20/6/2016

CH. 1 Introduction

* Digital Logic

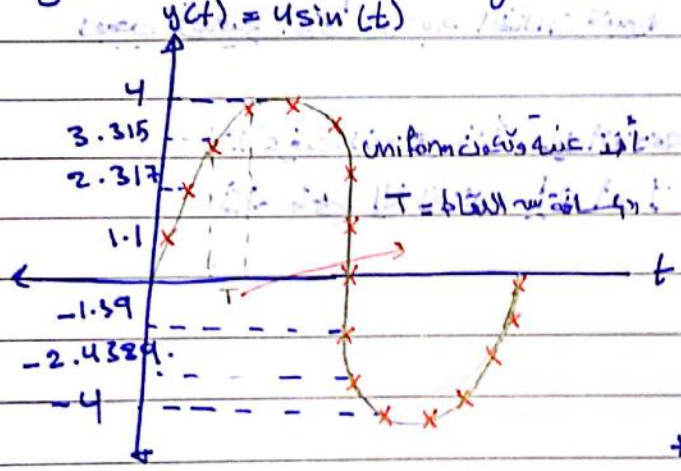
↳ Digital ⇒ Digit ⇒ Finger (Latin) ⇒ Finite and discrete
↳ Logic ⇒ Two values / possibilities.

* Digital logic deals with circuits that operate on two-valued elements (Binary)

* Binary values → True (T) High (H) 1 on
→ False (F) Low (L) 0 OFF

In such circuits (systems), everything is represented using these two values.

* Why Discrete and Binary systems?



* Infinite points?! مشكلة
لا نستطيع تخزين عدد لانهائي من البيانات في الذاكرة لذلك حل هذه المشكلة أخذ عينة (sampling) كهيك نحل المشكلة ونوصل finite number of points.

* infinite digits مشكلة
والتخلص من هذه المشكلة نأخذ عدد الـ digits وحسب منواجه مشكلة الـ error وتسمى هذه العملية التكميم (Quntization)

Sample Value	Qunt. Value	Quntized Binary
-4.0	-4	000
-3.2	-3	001
-2.3	-2	010
-1.0	-1	011
1.3	1	100
2.3	2	101
3.5	3	110
3.9	3	110

Multivalued Logic — Binary Logic (ثنائي)

* Binary are less complex
* Binary systems have better noise immunity (مقاومة)

noise

A (-4) ⟶ B (-3.6) هنالك خطأ بسبب الـ noise في هذا النظام
يصعب تصحيح الخطأ لوجود احتمالات كبيرة

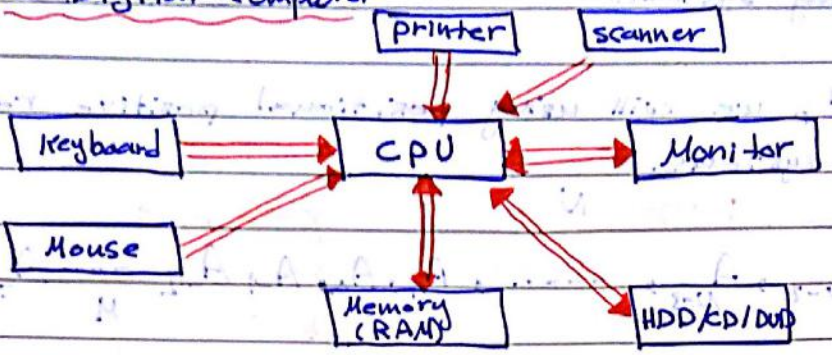
A 0 — noise — B 1
0 — 0

إذا عرفت انه في خطأ سهل تصحيحه
لأنه ما في إلا احتماليه يا 0 أو 1

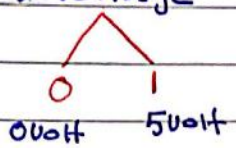
Scanned by CamScanner

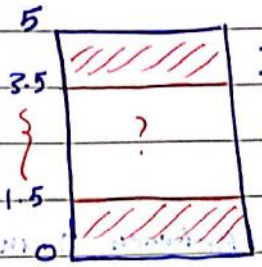
21/6/2016

1.2 The Digital computer

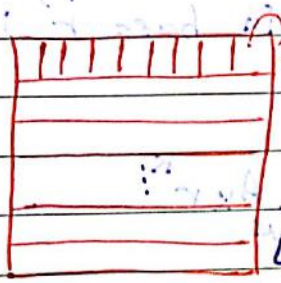


CPU ⇒ Voltage

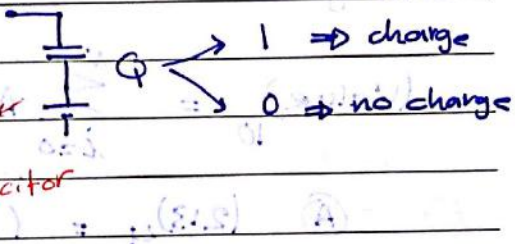




* Memory

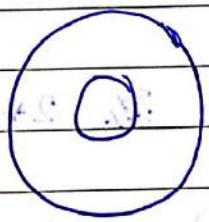


Memory ⇒ charge



Hard Disk
HDD

* Disk ⇒ Magnatic

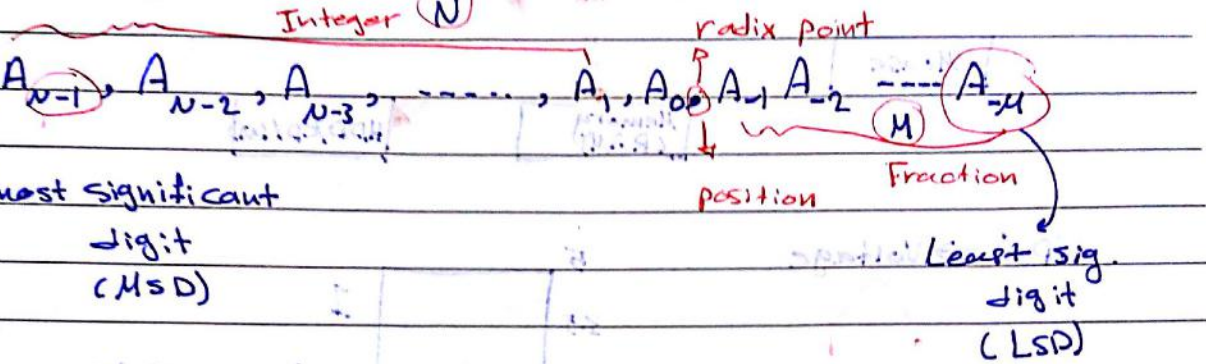


CD / DVD ⇒ light

21/6/2016

1.3 Numbering systems

* In general, we will using positional positive radix (base) numbering systems.



* where is a value between 0 and r-1; with 'r' being the base.

* The decimal value for a number in base 'r' is computed using.

$$(\text{Value})_{10} = \sum_{i=0}^{N-1} A_i * r^i + \sum_{K=-M}^{-1} A_K r^K$$

(Integer; Fraction)

Ex:- (A) $(213)_4 = (\;??\;)_{10}$

position: 2 1 0; base 4; base 10

$= 2*4^2 + 1*4^1 + 3*4^0 = (39)_{10}$

(B) $(55.3)_6 = (\;??\;)_{10}$

position: 1 0 -1

$= 5*6^1 + 5*6^0 + 3*6^{-1} = (35.5)_{10}$

21/6/2016

* Binary Numbering system

⇒ Base = r = 2

⇒ Digits ∈ {0, 1}

$(1011.01)_2$

Binary Digit = Bit

Example 2:- (A) $(10110)_2 = (??)_{10}$

$= 1*2^4 + 0*2^3 + 1*2^2 + 1*2^1 + 0*2^0 = (22)_{10}$

* Octal Numbering system

⇒ r = 8 ⇒ base 2^3

⇒ digit ∈ {0, 1, 2, 3, 4, 5, 6, 7}

* Hexadecimal Numbering system

⇒ r = 16 = 2^4

⇒ digit ∈ {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15}

A B C D E F

$(154)_{16}$

∵ F.

Example 3:- (A) $(1453)_{10} = (??)_{10}$

$1*16^3 + 4*16^2 + 5*16^1 + 3*16^0 = (5205)_{10}$

(B) $(F16)_{16} = (??)_{10}$

$15*16^2 + 1*16^1 + 6*16^0 = (3859)_{10}$

21/6/2016

© $(2D.8)_{16} = (??)_{10}$

$2*16 + 13*16^{0} + 8*16^{-1} = (45.5)_{10}$

Example 4 :-

0 - 16

- Decimal
- Binary
- Octal
- Hex

Decimal	Binary	octal	Hex
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	10000	20	10
			(16)

22/6/2016

1.4 Conversion between numbering systems :-

* conversion from base "r" to base 10.

$$(\text{Value})_{10} = \sum_{k=0}^{N-1} A_k r^k + \sum_{i=-M}^{-1} A_i r^i$$

* conversion from base "10" to base "r"

⇒ for Integer part, use repeted division on successive quotients and same the remainders. when then quotient is zero, stop and read the remainder from last to first

Example 5 :-

(A) $(37)_{10} = (??)_2$

$= (10010\,1)_2$ (5 4 3 2 1 0)

37	
18	1 ↑ LSD (LSB)
9	0
4	1
2	0
1	0
0	1 MSD (MSB)

كلمة منقسم على (2)

(B) $(153)_{10} = (??)_8 = (231)_8 = 231_{(8)}$ (2 1 0) octal

153	
19	1 ↑ LSD
2	3
0	2 MSD

كلمة منقسم على 8

22/6/2016

ⓒ $(53)_{10} = (??)_{3} = (1222)_{3}$ (3 2 1 0)

53	
17	2 ↑ LSD
5	2
1	2
0	1 ↓ MSD

... digits بتزيد ال base كل ما قلت ال OR

Example ⓑ $(53)_{10} = (??)_{16} = (35)_{16} = 0X35$ (Hex.)

= 35H → Hex.

53	
3	5 ↑
0	3

⇒ for the fraction part, use repeated multiplication on the fraction part. and save the Integer result. when the fraction is zero, stop and read the saved values from first to last.

Example (6)

Ⓐ $(0.6875)_{10} = (??)_{2} = (0.1011)_{2}$

MSD

0.6875 * 2 = 1.375

0.375 * 2 = 0.75

0.75 * 2 = 1.5

0.5 * 2 = 1.0 LSD

22/6/2016

(B) $(0.35)_{10} = (??)_4 = (0.11212)_4$

$0.35 * 4 = (1).4$
$0.4 * 4 = (1).6$
$0.6 * 4 = (2).4$
$0.4 * 4 = (1).6$
$0.6 * 4 = (2).4$

* في هذه الحالة بحدد عدد الـ digit يلي ← ما رح أوصل للـ fraction ← (0)
ثانياً إياهم ...

(C) $(7562.45)_{10} = (??)_8 = (16612.3463146)_8$

نأخذ الـ Integr ونقسم ← على 8

7562	
945	2 ↑ LSD
118	1
14	6
1	6
0	1 MSD

$0.45 * 8 = (3).6$
$0.6 * 8 = (4).8$
$0.8 * 8 = (6).4$
$0.4 * 8 = (3).2$
$0.2 * 8 = (1).6$
$0.6 * 8 = (4).8$
$0.8 * 8 = (6).4$

Five Apple

22/6/2016

Conversion from r1 to r2

r1 → Base 10 → r2

Example (7)

$(10110)_2 = (??)_5$

نحول Base 10 → $1*2^4 + 0*2^3 + 1*2^2 + 1*2^1 + 0*2^0 = (22)_{10} = (42)_5$

بالقسمة على (5)

22	
4	2
0	4

Example (8) :- A $(110011)_2 = (??)_{16}$ → 2^4 → 1→4 : الخانات

(B) $(110011)_2 = (??)_8$ → 2^3 → 1→3 : الخانات

(A) $(110011)_2 = (51)_{10} = (33)_{16}$

0011 0011.

$(3\ 3)_{16}$

(B) $(110011.)_2 = (51)_{10} = (63.)_8$

11 00 11.

$(6\ 3)_8$

(C) $(12F.D)_{16} = (??)_2$

0001, 0010, 1111, 1101

$= (0001\,0010\,1111.1101)_2$

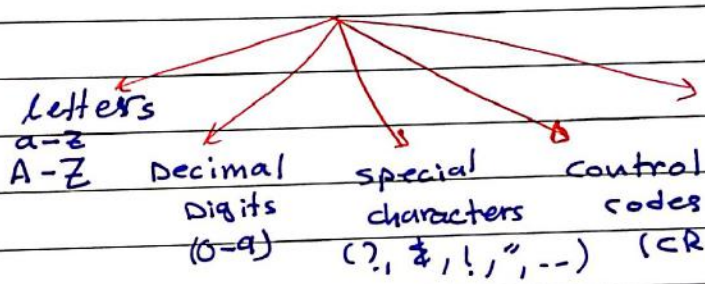
1 2 F D

(D) $(1220)_3 = (??)_9$

$(56)_9$

1.5 Alphanumeric codes :-

ASCII ⇝ American standard code for information interchang.



* ASCII uses 7 bits to encod all these characters.

* $2^7 = 128$ items can be encoded

optical shaft encodar
كما اقرأ في الكتاب

25/6/2016

* Unicode :-

⇒ 16 bits ⇒ $2^{16} = 2^{10} * 2^{6} = 64 * 1024$ (Kilo)

It's an extension to (ASCII)

$A = (41H)_{ASCII} = (0041H)_{unicode}$

$\alpha = (03B1.H)_{unicode}$ ← لا يوجد لها مقابل في الـ ASCII

$= ((0000\ 0011\ 1011\ 0001))_{2\ unicode}$

$\div = (.0638H)$

* Gray code :-

⇒ non numeric code ← لا يستخدم لتمثيل الأرقام

⇒ the representation between two adjacent codes differs by one bit change.

2-bit

	Binary	Random	Gray code
ORANGE	00	11	00
APPLE	01	00	01
MEION	10	10	11
STRAWBERRY	11	01	10

25/6/2016

* Parity code

⇒ extra bit(s) that are appended to the original data in order to facilitate error detection.

⇒ parity could be odd or even.

	Even	odd
1011	11011	01011
1001	01001	11001

مجموع ال (1) عدد فردي ← (odd)

مجموع ال (1) لازم يكون عدد زوجي (Even)

A → B

parity bit ← ①1011 ⟶ 10011 → لا يصحح الخطأ ولكن يكشفه
error

11011 ⟶ 00001

1.6 Binary Coded Decimal Code (BCD)

$(27)_{10} = (11011)_2 = (0010\ 0111)_{BCD}$ — 0010 = 2, 0111 = 7

0 – 9

0000, 0001, 0010, 0011
0100, 0101, 0110, 0111
1000, 1001

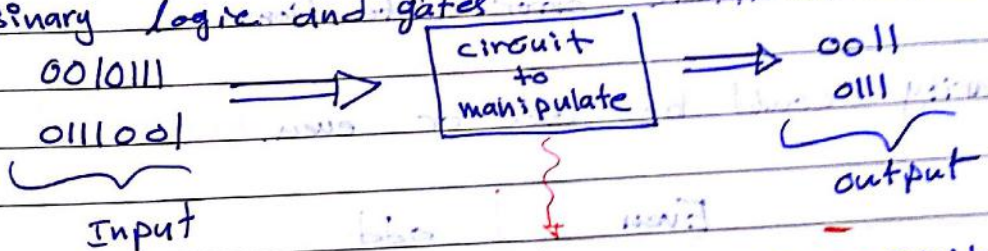
$(135)_{10} = (1000\ 0111)_2 = (0001\ 0011\ 0101)_{BCD}$ — 0001 = 1, 0011 = 3, 0101 = 5

26/6/2016

Chapter "2"

Combinational Logic circuits :-

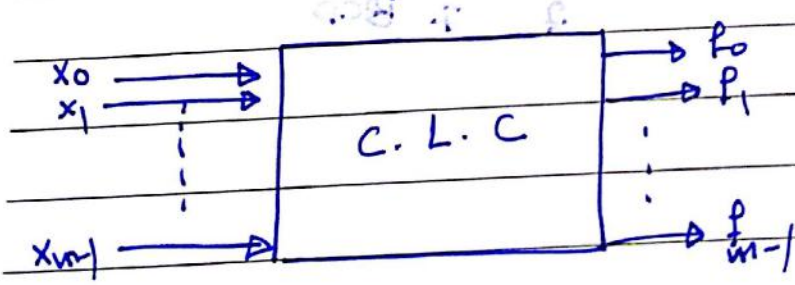
2.1 Binary Logic and gates



Transistors + resistors + wires = Digital logic circuit

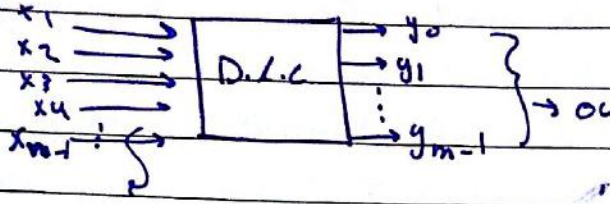
Binary ⇒ logic circuits → simple (Basic) ⇒ perform basic Logic operations
↳ Logic gates
→ complex
↳ combination of basic operations

⇒ Logic circuits → combinational (output depends on input only)
→ sequential → (output depends on input and state).



f_i = function ($x_0, x_1, \ldots x_{n-1}$)

27/6/2016



* D.L.C → C.L.C $y_i = F(x_1, x_2, x_3, \ldots x_{n-1})$
→ S.L.C $y_i = F(\text{input} + \text{state})$

* Binary logic ;

⇒ describes the operation of D.L.C in mathematical form.

⇒ it deals with binary variable only.

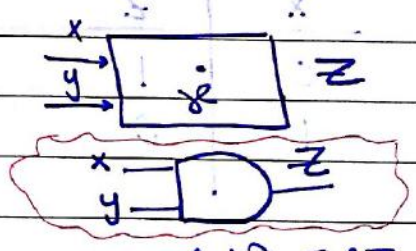
x, y, ALARM } 0 OFF F
1 on T

⇒ Basic Binary operations ;—

1. AND $z = x \odot y = xy$

and

0 = 0 . 0	
0 = 0 . 1	
0 = 1 . 0	
1 = 1 . 1	



AND GATE

x	y	z
0	0	0
0	1	0
1	0	0
1	1	1

Truth Table

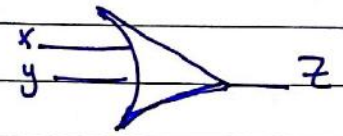
27/6/2016

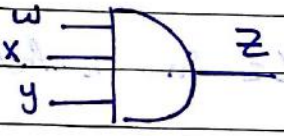
[2] OR

OR

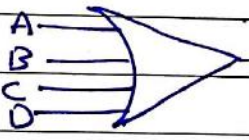
$Z = X + Y$

x	y	z
0	0	0
0	1	1
1	0	1
1	1	1





$Z = w.x.y$



$F = A+B+C+D$

[3] NOT / Complement

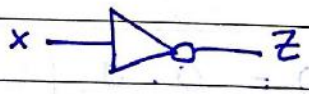
not

$Z = \overline{X} = \sim X = X'$

not

not

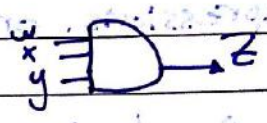
X	Z
0	1
1	0



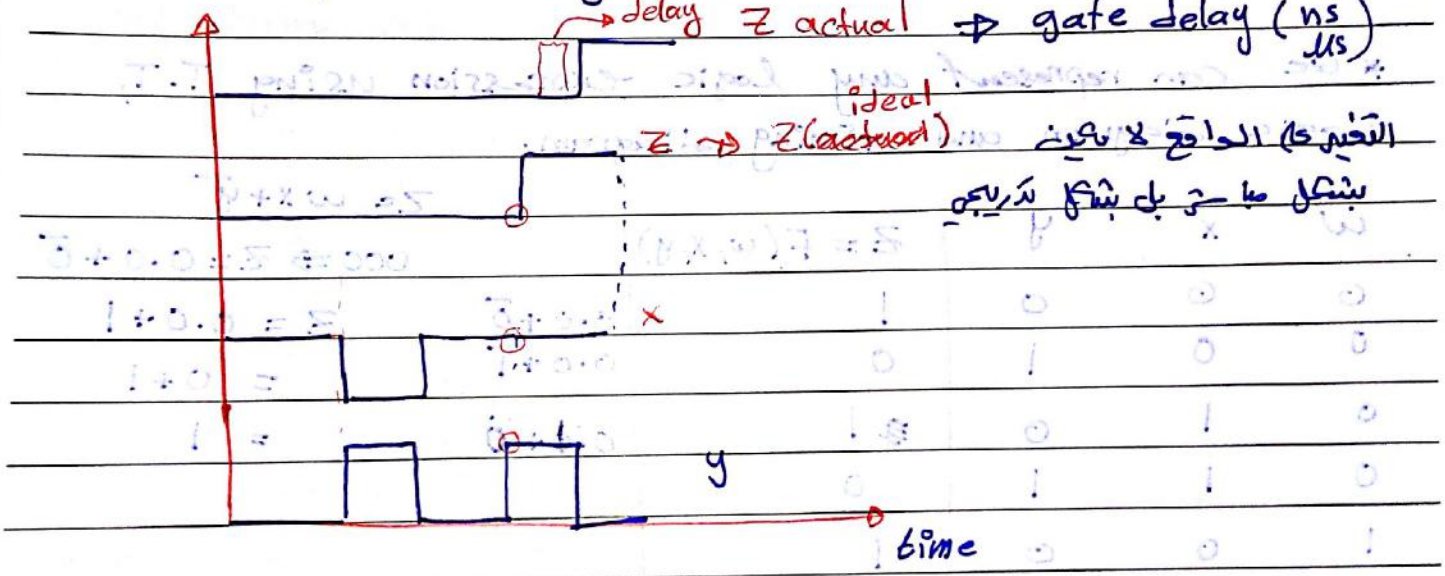
27/6/2016

$2^3 = 8$

w	x	y	z
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

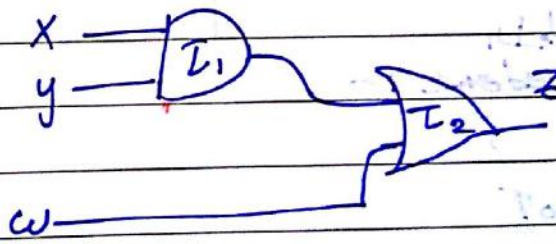


AND gate timing diagram



التغيير في الواقع لا يكون بشكل مباشر بل بشكل تدريجي

Gate delay :- It's the difference between the (Actual) time and the (ideal) time.



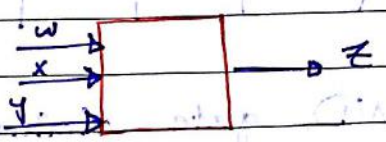
27/6/2016

2.2 Boolean Algebra.

⇒ is used to manipulate Boolean / Binary / Logic expressions or function.

⇒ A Logic expression is a mix of Binary variables, logic operations and the constants 0 and 1.

$$Z = F(w, x, y) = wx + \bar{y}$$



* we can represent any logic expression using T.T, Logic diagram and timing diagram.

w	x	y	$Z = F(w, x, y)$
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

$Z = wx + \bar{y}$

000 ⇒ $Z = 0.0 + \bar{0}$

$0.0 + \bar{0}$

$0.0 + \bar{1}$

$0.\phi + \bar{0}$

$Z = 0.0 + 1$

$= 0 + 1$

$= 1$

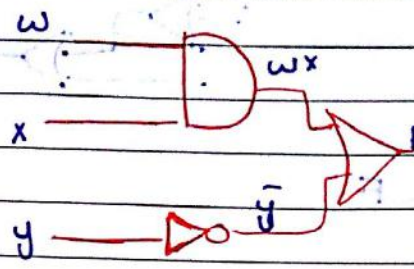
* precedence :-

1. ( )
2. NOT
3. AND
4. OR

Five Apple

27/6/2016

$F = wx + \bar{y}$



Example :- $F(A, B, C, D) = A + \bar{B}D + \bar{C}$

MSB — LSB

(a) Derive the T.T

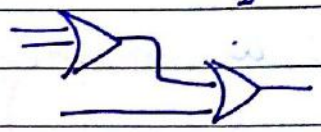
A	B	C	D	F	
0	0	0	0	1	$0 + \bar{0}.0 + \bar{0}$
0	0	0	1	1	
0	0	1	0	0	
0	0	1	1	1	
0	1	0	0	1	
0	1	0	1	1	
0	1	1	0	0	
0	1	1	1	0	$0 + \bar{1} + 1 + \bar{1}$
1	0	0	0	1	
1	0	0	1	1	
1	0	1	0	1	
1	0	1	1	1	
1	1	0	0	1	
1	1	0	1	1	
1	1	1	0	1	
1	1	1	1	1	

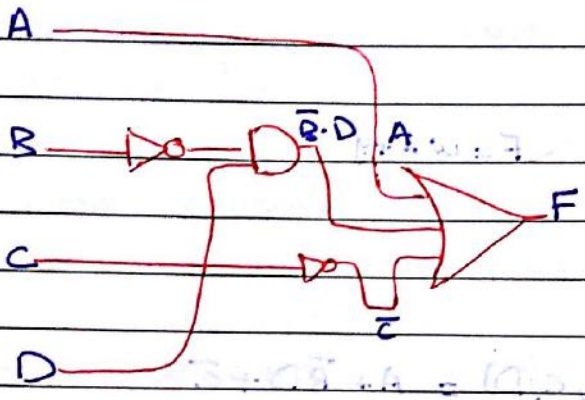
Five Apple

27/6/2016

(b) Draw the Logic Diagram

مثال توضيحي 3variable لدالة بولية



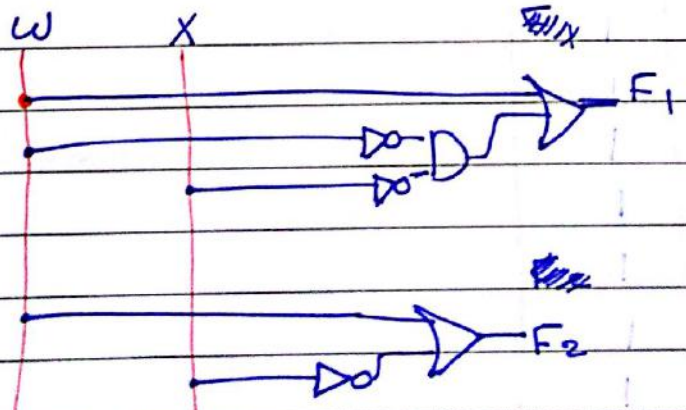


Example $F_1(w,x) = \bar{w}\bar{x} + w$

$F_2(w,x) = w + \bar{x}$

w	x	F_1	F_2
0	0	1	1
0	1	0	0
1	0	1	1
1	1	1	1

same function



27/6/2016

* A logic expression may have multiple algabraic representation (Logic diagrame); however, it has one T.T

27/6/2016

* A logic expression may have multiple algabraic representation (Logic diagrame); however, it has one T.T

Boolean identities :-

(1) $\{ X+0 = X \quad X.0 = 0 \}$ $\{ X+1 = 1 \quad X.1 = X \}$ existance of 1 or 0

(2) $\{ X + X = X \quad X.X = X \}$ Idempotance

(3) $\{ \bar{\bar{X}} = X \}$ Involution

(4) $X + Y = Y + X$ $X.Y = Y.X$ Commutative

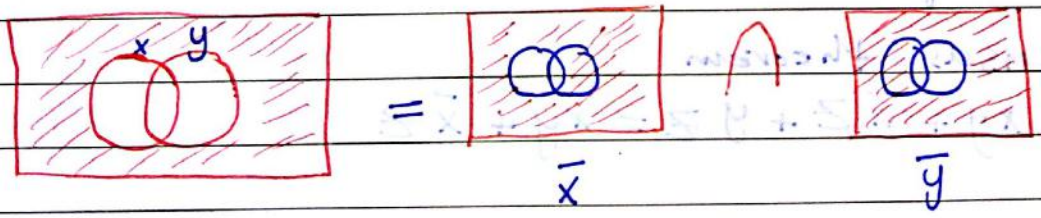
(5) $(X+Y)+Z = X+(Y+Z)$ $(X.Y).Z = X.(Y.Z)$ Associative

(6) $X.(Y+Z) = X.Y + X.Z$ Distributive

$X + Y.Z = (X+Y)(X+Z)$ ⇒ very Important

(7) $X + \bar{X} = 1$ $X.\bar{X} = 0$ Existance of complemat

(8) $\overline{X+Y} = \bar{X}.\bar{Y}$ $\overline{X.Y} = \bar{X} + \bar{Y}$ Demorgan Law.



28/6/2016

مثال

$\overline{(x+y)\cdot z} = \overline{(x+y)} + \overline{z}$

$= (\bar{x}\cdot\bar{y}) + \bar{z}$

$\overline{\overline{w+z}} = w+z$

* Boolean theorems :-

① * Simplification theorem

$x + \bar{x}y = x+y$

proof :-

$(x+\bar{x})(x+y)$ Distribution

$1\cdot(x+y)$ Existance of complement

$(x+y)$ Existance of 1

② * Absorption theorem

$x + x\cdot y = x$

proof :- ~~$(x+x)(x+y)$~~

$x\cdot(1+y)$ Dist.

$x\cdot 1$

x

③ Minimization theorem

$xy + \bar{x}y = y$

proof :-

$(x+\bar{x})\cdot y$

$1\cdot y$

y

④ Consensus theorem

$xy + \bar{x}z + yz = xy + \bar{x}z$

proof → in the Book.

28/6/2016

Example :- Show that

$$F(x,y) = xy + \bar{x}y + x\bar{y} + \bar{x}\bar{y} = 1$$

$$= xy + x\bar{y} + \bar{x}y + x\bar{y}$$

$$= x(y+\bar{y}) + \bar{x}(y+\bar{y})$$

$$= x + \bar{x}$$

$$= 1$$

x	y	F
0	0	1
0	1	1
1	0	1
1	1	1

Example :- show that

$$ABC + \bar{A}\bar{C} + A\bar{C} = AB + \bar{C}$$

$$ABC + (\bar{A}+A)\bar{C}$$

$$ABC + \bar{C}$$

$$(AB+\bar{C}).(C+\bar{C}) \Rightarrow AB + \bar{C}$$

$$ABC + AB\bar{C} + \bar{C}.C + \bar{C}.\bar{C}$$

$$\vdots$$

* The complement of a function

$$F(w,x,y) \Rightarrow \overline{F(w,x,y)} \; ??$$

⇒ In the truth table, the complement $\bar{F}$ replaces the 0's of F with 1s and the 1s with 0s

x	y	F	$\bar{F}$
0	0	1	0
0	1	0	1
1	0	1	0
1	1	1	0

⇒ using algebra; we apply Demorgan law. ⟹

23/6/2016

Exercise 5

$F(A,B,C) = AB + \bar{C}$

$\bar{F}(A,B,C) = ??$

ABC	F	$\bar{F}$

$\bar{F}(A,B,C) = \overline{(AB+\bar{C})}$

$= \bar{A}\bar{B} \cdot \bar{\bar{C}}$

$= (\bar{A}+\bar{B}) \cdot C$

$\bar{F} = \bar{A}C + \bar{B}C$

Example: $F(A,B,C,D) = AB(A+\bar{C}) + \bar{B}(\bar{A}D+C)$

$\bar{F}(A,B,C,D) = ??$

$= \overline{AB(A+\bar{C}) + \bar{B}(\bar{A}D+C)}$

$= \overline{AB(A+\bar{C})} \cdot \overline{\bar{B}(\bar{A}D+C)}$

$= [\bar{A}\bar{B} + \overline{(A+\bar{C})}] \cdot [\bar{\bar{B}} + \overline{(\bar{A}D+C)}]$

$= [(\bar{A}+\bar{B}) + \bar{A}\cdot\bar{\bar{C}}] \cdot [B + \overline{\bar{A}D}\cdot\bar{C}]$

$= [\bar{A}+\bar{B} + \bar{A}\cdot C] \cdot [B + (\bar{\bar{A}}+\bar{D})\cdot\bar{C}]$

$= [\bar{A}(1+C) + \bar{B}] \cdot [B + (A+\bar{D})\cdot\bar{C}]$

$= [\bar{A}+\bar{B}][B + A\bar{C} + \bar{C}\bar{D}]$

$= \bar{A}B + \bar{A}\cdot A\bar{C} + \bar{A}\bar{C}\bar{D} + \bar{B}\cdot B + \bar{B}A\bar{C} + \bar{B}\bar{C}\bar{D}$

($A\bar{C}$ and $\bar{B}\cdot B$ are 0)

Sum of product.

29/6/2016

2.3 Standard Forms

* Dual of a function

$X \oplus 0 = X$, $X \odot 1 = X$ ⇒ Replace $+ \rightleftarrows \cdot$, $0 \rightleftarrows 1$

* standard forms :-

* Logic expressions may have different forms, but the same T.T

* we need an approach to simplify comparison between Logic functions

* This approach should 1. correspond to the T.T
2. facilitate simplification

Forms

- standard
  - sum of product (SOP)
  - product of sum (POS)
- non-standard

$A.(B+\bar{A}C)+\bar{A}B(C+\bar{B})$

$F(A,B,C) = AB + \bar{B}C + ABC$

$(A+\bar{B}).(\bar{B}+C)$

$(\bar{A}+\bar{C})$

⇓ ⇓

- Sum of minterms (SOM)
- product of Maxterms (POM)

⇒ canonical forms

$F(A,B,C) = \bar{A}BC + A\bar{B}\bar{C} + ABC$

$(A+B+\bar{C}).(\bar{A}+B+\bar{C})$

Five Apple

29/6/2016

* Sum of minterms (SOM)

	x	y	F	
m_0	0	0	0	
m_1	$\bar{0}$	1	1	$\bar{0}.1$
m_2	1	0	0	+
m_3	1	1	1	1.1

يتم اخذ الـ 1s التي موجودة عندنا لكي نقدر نكتب الـ expression

MSV ↑ $\bar{0}.1 + 1.1$

$F(x, y) = \bar{x} \cdot y + x \cdot y$

LSV — minterm — minterm

$F(x, y) = m_1 + m_3$

$F(x, y) = \sum_m (1, 3)$

miterm :- it's contain all the variable of the function

Example :- ③ 1s يعني ③ minterms

	A	B	C	F	
m_0	0	0	0	0	
m_1	$\bar{0}$	$\bar{0}$	1	1	$\bar{A}.\bar{B}.C$
m_2	0	1	0	0	
m_3	0	1	1	0	+
m_4	1	$\bar{0}$	$\bar{0}$	1	$A.\bar{B}.\bar{C}$
m_5	1	0	1	0	+
m_6	1	1	$\bar{0}$	1	$A.B.\bar{C}$
m_7	1	1	1	0	

$F(A,B,C) = \bar{A}\bar{B}C + A\bar{B}\bar{C} + ABC$

$= m_1 + m_4 + m_6$

$= \sum_m (1, 4, 6)$

29/6/2016

Example :-

$F(\omega, x, y) = \sum_m (2, 5, 7)$

(a) write the Logic expression of F as SOM

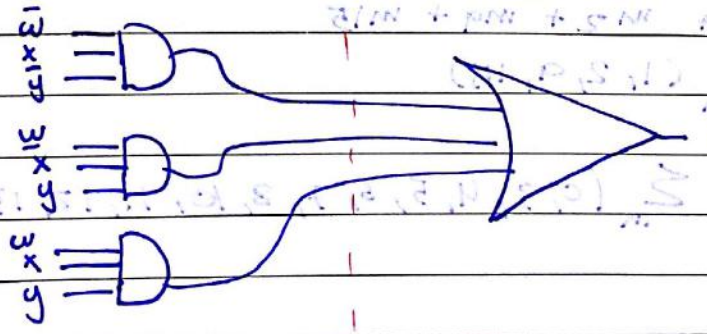
$F(\omega, x, y) = \underset{010}{m_2} + \underset{101}{m_5} + \underset{111}{m_7}$

$F(\omega, x, y) \quad \bar{\omega} x \bar{y} + \omega \bar{x} y + \omega x y$

(b) T.T

A	B	C	F
0	0	0	
0	0	1	1 m_2
0	1	0	
0	1	1	1 m_5
1	0	0	
1	0	1	1 m_7
1	1	0	
1	1	1	

(c) Draw the Logic Diagram



product sum → two level circuit

29/6/2016

Example

(a)

	w	x	y	z	F	$\bar{F}$
0					0	1
1	$\bar{0}$	$\bar{0}$	$\bar{0}$	1	(1) m_1	0
2	$\bar{0}$	$\bar{0}$	1	$\bar{0}$	(1) m_2	0
3					0	1
4					0	1
5					0	1
6					0	1
7					0	1
8					0	1
9	1	$\bar{0}$	$\bar{0}$	1	(1) m_9	0
10					0	1
11					0	1
12					0	1
13					0	1
14					0	1
15	1	1	1	1	(1) m_{15}	0

$$F(w,x,y,z) = \bar{w}\bar{x}\bar{y}z + \bar{w}\bar{x}y\bar{z} + w\bar{x}\bar{y}z + wxyz$$
$$= m_1 + m_2 + m_9 + m_{15}$$
$$= \sum_m (1, 2, 9, 15)$$

(b) $\overline{F(w,x,y,z)} = \sum_m (0, 3, 4, 5, 6, 7, 8, 10, 11, 12, 13, 14)$

5/7/2016

ex:- $F(x,y,z) = \bar{x}y\bar{z} + \bar{x}yz + \bar{x}\bar{y}z$

T.T ??

$= m_6 + m_3 + m_1$

$= \sum_m (1, 3, 6)$

x	y	z	F
			0
			1 m_1
			0
			1 m_3
			0
			0
			1 m_6
			0

* converting From Logic expression to SOM.

ex:- $F(x,y,z) = xy + \bar{x}z$ ⇒ SOP

write the SoM expression for F ?!

$F(x,y,z) = xy \cdot 1 + \bar{x}z \cdot 1$

$= xy \cdot (z + \bar{z}) + \bar{x}(y + \bar{y})z$

$= xyz + xy\bar{z} + \bar{x}yz + \bar{x}\bar{y}z$ ⇒ Minterms

(1 1 1) (1 1 0) (0 1 1) (0 0 1)

$= m_7 + m_6 + m_3 + m_1$

$= \sum_m (1, 3, 6, 7)$

3/7/2016

ex:- $F(A,B,C,D) = A(A\bar{C}+\bar{B}) + A\bar{B}(C+A)$

$= A\bar{C} + A\bar{B} + A\bar{B}C + A\bar{B}$ ⇒ SOP

$= A(B+\bar{B})\bar{C} + A\bar{B}(C+\bar{C}) + A\bar{B}C$

$= AB\bar{C} + A\bar{B}\bar{C} + A\bar{B}C + A\bar{B}\bar{C} + A\bar{B}C$

$= m_6 + m_4 + m_5 + m_4 + m_5$

$= \sum_m (4,5,6)$

* product of Maxterms :-

	x		y	F		
M_0	0	+	0	0	0+0	$(x+y)$
M_1	0		1	1		
M_2	$\bar{1}$	+	0	0	$\bar{1}+0$	$(\bar{x}+y)$
M_3	1		1	1		

$F(x,y) = (x+y)\cdot(\bar{x}+y)$

maxterm maxterm

$= M_0 \cdot M_2$

$= \Pi_M (0,2)$

Example:-

x		y		z	F		
0	+	0	+	0	0	0+0+0	$(x+y+z)$
0		0		1	1		
0		1		0	0	$0+\bar{1}+\bar{1}$	$(x+\bar{y}+\bar{z})$
0	+	1	+	1	1		
1		0		0	0	$\bar{1}+0+\bar{1}$	$(\bar{x}+y+\bar{z})$
$\bar{1}$	+	0	+	$\bar{1}$	0	$\bar{1}+\bar{1}+0$	$(\bar{x}+\bar{y}+z)$
$\bar{1}$	+	$\bar{1}$	+	0	1		
1		1		1			

⇒

3/7/2016

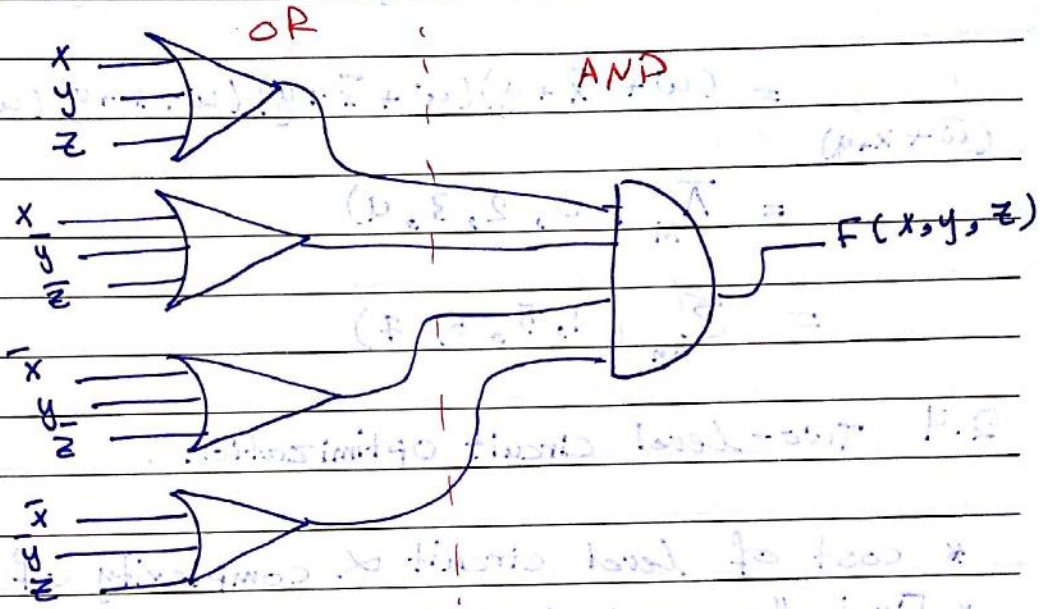
(a) POM ??

$F(x,y,z) = (x+y+z).(x+\bar{y}+\bar{z}).(\bar{x}+y+\bar{z}).(\bar{x}+\bar{y}+z)$

$= M_0 . M_3 . M_5 . M_6$

$F(x,y,z) = \Pi_M (0,3,5,6)$ ← الصفوف اللي بتجبرها عند 0's

(b) Draw the POM expression of F ?!



(c) $F(x,y,z) = \sum_m (\; ? \;)$

$= \sum_m (1,2,4,7)$ الصفوف اللي بتجبرها عند الـ 1's

$= \bar{x}\bar{y}z + \bar{x}y\bar{z} + x\bar{y}\bar{z} + xyz$

(d) $\overline{F(x,y,z)} = \sum_m (0,3,5,6)$

(E) $\overline{F(x,y,z)} = \Pi_m (1,2,4,7)$

Five Apple

3/7/2018

* converting From Logic expression to POM.

ex:- $F(w, x, y) = wx + \bar{x}y$

$= (wx + \bar{x})(wx + y)$ (dist) توزيع

$= (w + \bar{x})(x + \bar{x})(w + y)(x + y)$

$= (w + \bar{x}) \cdot (w + y)(x + y)$ POS

$= (w + \bar{x} + y \cdot \bar{y})(w + \bar{x} \cdot \bar{x} + y)(w \cdot \bar{w} + x + y)$ توزيع

$= (w + \bar{x} + y)(w + \bar{x} + \bar{y})(w + x + y)(w + \bar{x} + y)(w + x + y)$

$(\bar{w} + x + y) \Rightarrow$ MaX terms

$= \Pi_M (0, 2, 3, 4)$

$= \Sigma_m (1, 5, 6, 7)$

2.4 Two-Level circuit optimization ... تحسين

* cost of Level circuit ∝ complexity of Logic expression
* Find the simplest expression!
* HOW??
  - Algebra (Hard)(not systematic)
  - K-Maps
  - other (QuiAD Mcgluski) ⇒ ANY # of variables prosramed

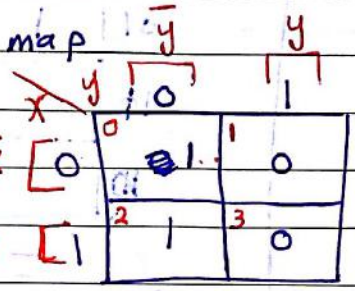
4/7/2016

* Karnaugh Map (K-Map)

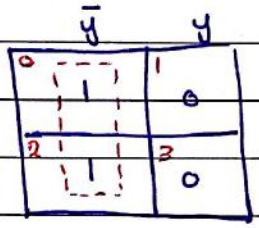
⇒ graphical representation of the Truth table that consists of a set of adjacent squares where each square corresponds to a minterm or row in the table.

⇒ 2-variable K-map

x	y	F
0	0	1
0	1	0
1	0	1
1	1	0



⇒



$F = m_0^{00} + m_2^{10}$

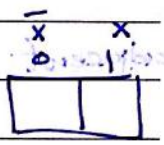
$= \bar{x}\bar{y} + x\bar{y}$ minimization

$= (\bar{x} + x)\bar{y}$

$= \bar{y}$

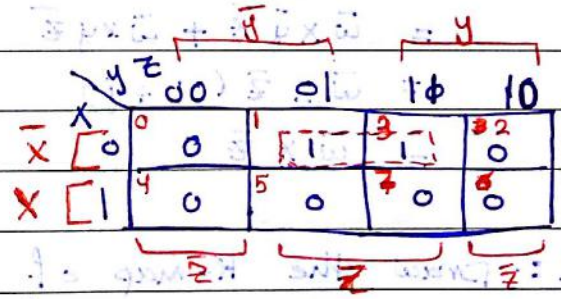
1-variable

x	F
0	1
1	x



⇒ 3-variable K-map

x	y	z	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0



$m_1 + m_3$

$\bar{x}\bar{y}z + \bar{x}yz$

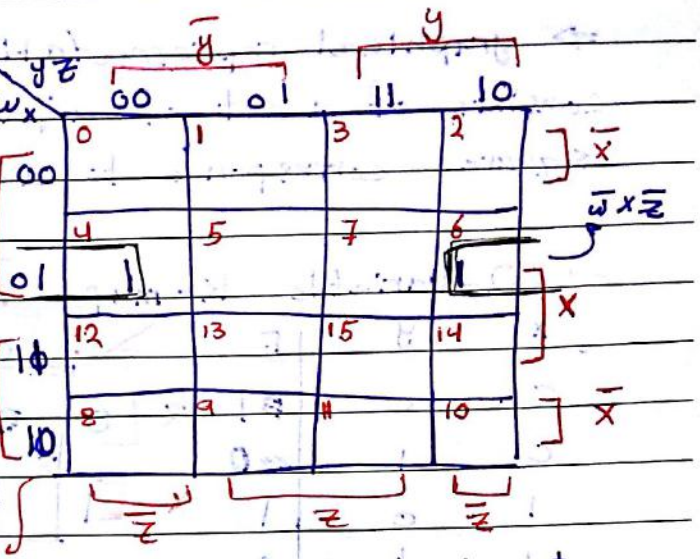
$\bar{x}z(\bar{y} + y)$

$\bar{x}z$

24/7/2016

⇒ 4 - variable $F(w, x, y, z) = \sum_m (4, 6)$

w	x	y	z	F



الترتيب حسب الـ Gray Code

shared side عبارة عن 0 و 8

0 → $w\bar{x}\bar{y}\bar{z}$

8 → $w\bar{x}\bar{y}\bar{z}$

adjacent هي ← 1 variable والاختلاف variable مستثنية بالباقي

⇒ $w\bar{x}\bar{z}$ $m_4 + m_6$

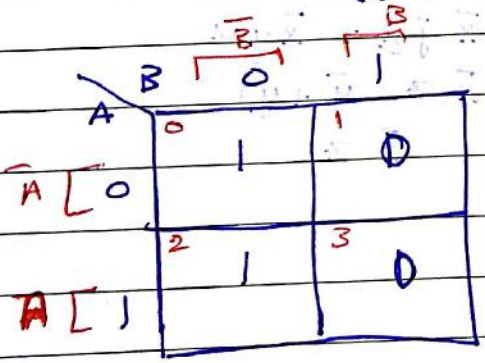
$= \bar{w}x\bar{y}\bar{z} + \bar{w}xy\bar{z}$

$= \bar{w}x\bar{z}(y + \bar{y})$

$= \bar{w}x\bar{z}$

Example :- Draw the K-map of

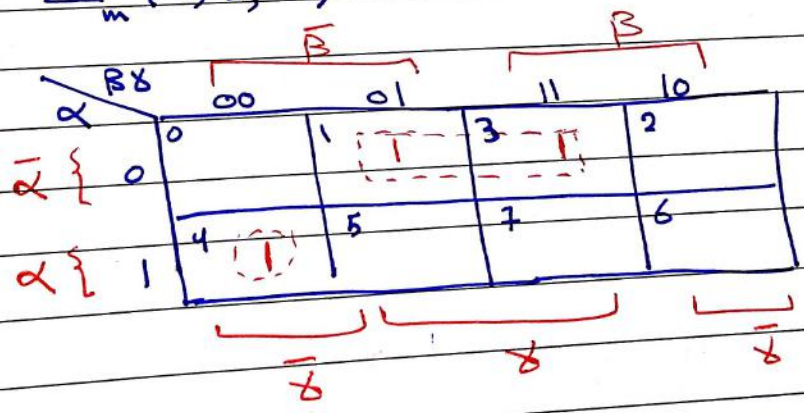
(a) $F(A, B) = \prod_M (1, 3) = \sum_m (0, 2)$



4/7/2016

(b) $F(\alpha, \beta, \gamma) = \alpha\bar{\beta}\bar{\gamma} + \bar{\alpha}\beta\gamma + \bar{\alpha}\bar{\beta}\gamma$

$= \sum_m (1, 3, 4)$



$F = \bar{\alpha}\gamma + \alpha\bar{\beta}\bar{\gamma}$

10/7/2016

* Map manipulation

⇒ we need to find the simplest expression ...

⇒ steps :-

1- Draw and fill the K-map ...

2- Identify all prime implicants (pI) and essential prime implicants (EpI).

3- write the sop logic expression such that :-

a) all EpI are included.

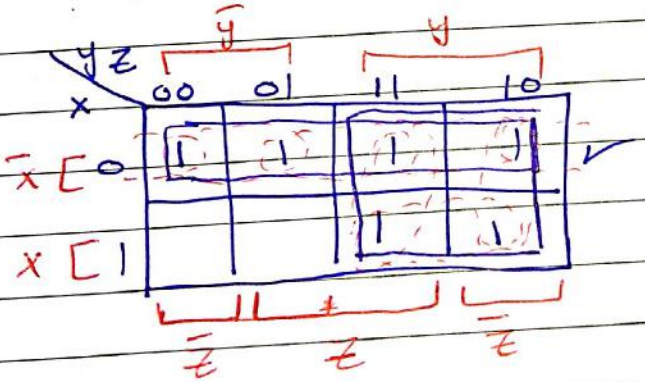
b) the minimum number of pI are used to cover all remaining 1s in the map ...

⇒ prime implicant (pI) :- It's product term (Largest rectangle) that contains power of 2 ones in the K-map.

⇒ (EpI) :- It's a pI with one or more 1's that are only included in this pI.

Example :-

$F(x,y,z) = \sum_m (0, 1, 2, 3, 6, 7)$



$\bar{x}\bar{y}\bar{z} + \bar{x}\bar{y}z + \bar{x}yz + \bar{x}y\bar{z}$

$\bar{x}\bar{y} + \bar{x}y$

$\bar{x}$ ✓

Five Apple

10/7/2017

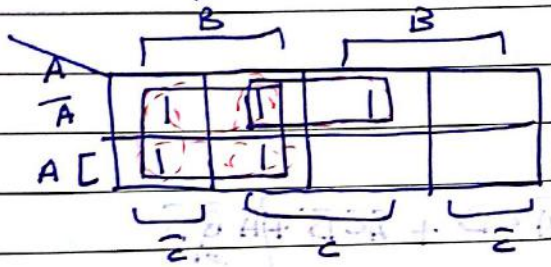
Write down the PI.?

PI ⇒ { $\bar{x}$ $\in$ PI

y $\in$ PI

$F(x, y, z) = \bar{x} + y$

Example:- $F(A, B, C) = \prod_M (2, 6, 7) = \sum_m (0, 1, 3, 4, 5)$

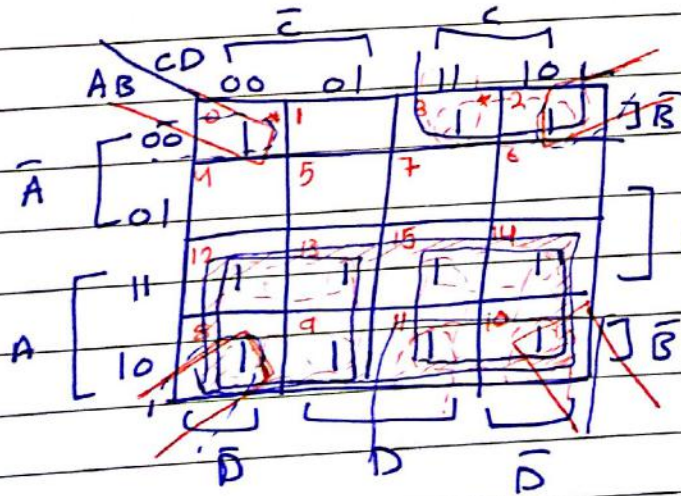


PI ⇒ $\bar{B}$ $\in$ PI

$\bar{A}C$ $\in$ PI

$F(A, B, C) = \bar{B} + \bar{A}C$

Example:- $F(A, B, C, D) = \sum_m (0, 2, 3, 8, 9, 10, 11, 12, 13, 14, 15)$



PI ⇒ A $\in$ PI ✓

$\bar{B}C$ $\in$ PI ✓

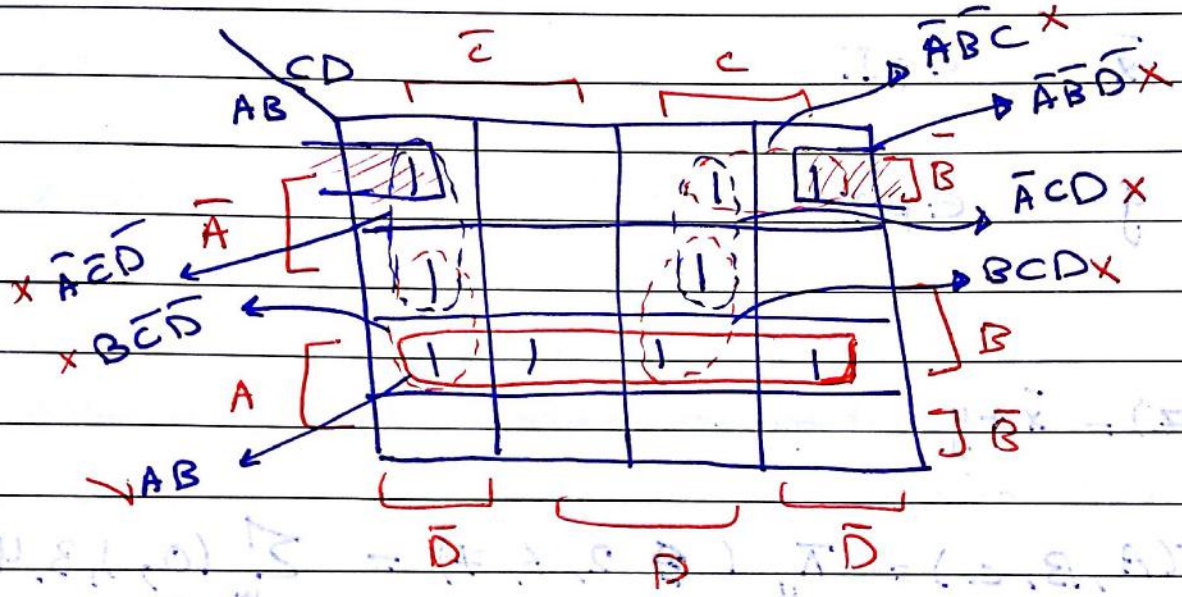
$\bar{B}\bar{D}$ $\in$ PI ✓

$F = A + \bar{B}C + \bar{B}\bar{D}$

Five Apple

10/7/2016

Example :- $F(A,B,C,D) = \sum_m (0,2,3,7,12,13,14,15)$



EPI :- AB

$\Rightarrow F(A,B,C,D) = AB + \bar{A}CD + \bar{A}C\bar{D} + \{\bar{A}\bar{B}\bar{D}, \bar{A}\bar{B}C\}$

11/7/2016

* product of sum optimization :-

Steps :-

1- Find $\bar{F}$ and fill the map...

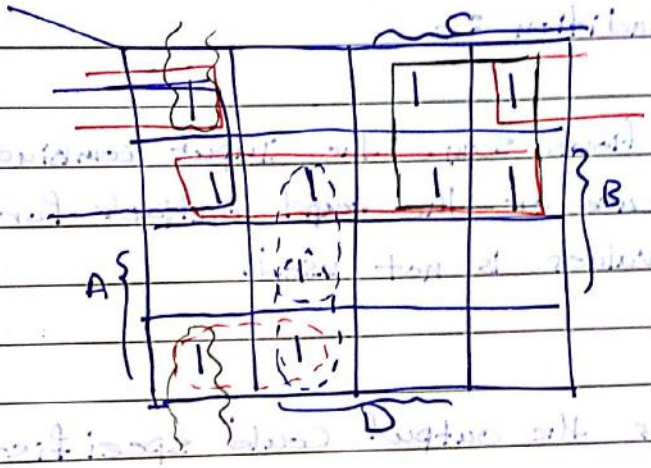
2- Simplify $\bar{F}$ as SOP...

3- Find F by complementing $\bar{F}$...

Example :-

$F(A,B,C,D) = \sum_m(0,2,3,4,5,6,7,8,9,13)$

(a) Find the sop expression of F.



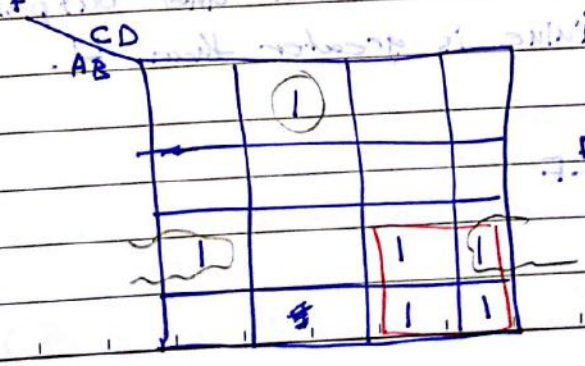
EPI :

$\bar{A}B$	X
$\bar{A}C$	✓
$\bar{A}\bar{D}$	X
$B\bar{C}D$	X
$A\bar{C}D$	X
$A\bar{B}\bar{C}$	X
$\bar{B}\bar{C}\bar{D}$	X

$F(A,B,C,D) = \bar{A}C + \bar{A}\bar{D} + B\bar{C}D + A\bar{B}\bar{C}$

(b) find the (POS) expression of f

$\bar{F} = \sum_m(1,10,11,12,14,15)$



$\bar{F}(A,B,C,D) = \bar{A}\bar{B}\bar{C}D + AB\bar{D} + AC$

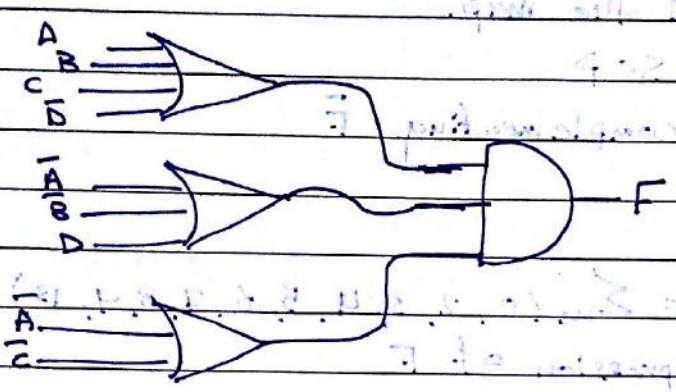
$F = \bar{\bar{F}} = \overline{\bar{A}\bar{B}\bar{C}D + AB\bar{D} + AC}$

Five Apple

11/7/2016

$= (A+B+C+\bar{D}) \cdot (\bar{A}+\bar{B}+D) \cdot (\bar{A}+\bar{C})$

Logic Diagram



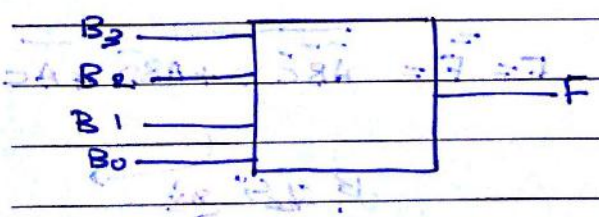
* Don't care condition :-

⇒ For some functions, some the Input combination might not occure or the output for some Input values is not used.

⇒ in such cases the output could specified as '0' or '1'.

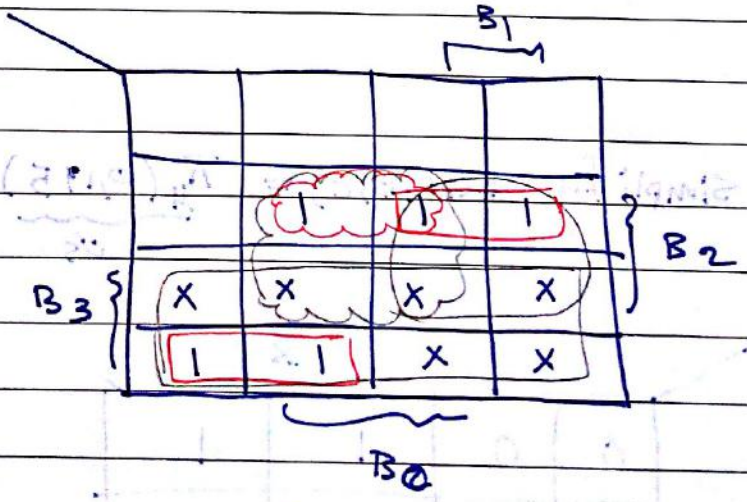
⇒ This is called 'Don't care' condition and is denoted by 'X'.

Example:- Determine the logic expression for a logic circuit that accepts a BCD number and outputs '1' if the Input BCD value is greater than 4.



11/7/2016

	B_3	B_2	B_1	B_0	F
BCD	0	0	0	0	0
	0	0	0	1	0
	0	0	1	0	0
	0	0	1	1	0
	0	1	0	0	0
	0	1	0	1	1
	0	1	1	0	1
	0	1	1	1	1
	1	0	0	0	1
	1	0	0	1	1
not used	1	0	1	0	X
	1	0	1	1	X
	1	1	0	0	X
	1	1	0	1	X
	1	1	1	0	X
	1	1	1	1	X



⇒ assuming all x's to 1 zeros.

$$F(B_3, B_2, B_1, B_0) = B_3\bar{B}_2\bar{B}_1 + \bar{B}_3 B_2 B_1 + \bar{B}_3 B_2 B_0$$

Five Apple

11/7/2016

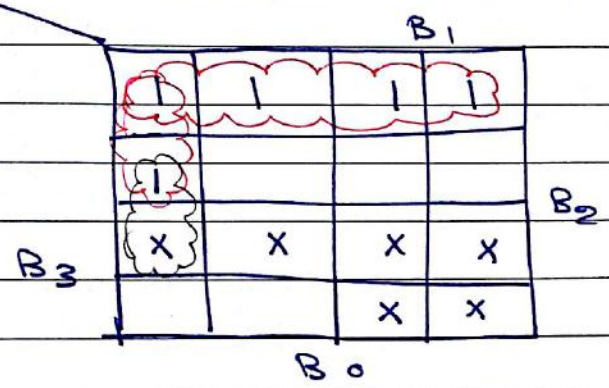
assuming all x's to be 1's

$$F = B_3 + B_2B_1 + B_2B_0$$

Note:- when simplifing the functions, we don't need to cover all the don't care

* pos of F??

$\bar{F}$

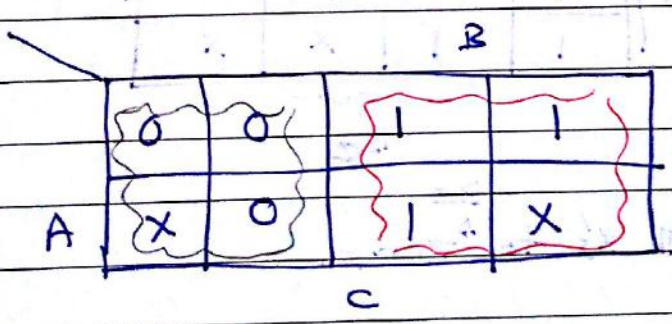


$$\bar{F} = \bar{B}_3\bar{B}_2 + \{B_2\bar{B}_1\bar{B}_0 \;\; \bar{B}_3\bar{B}_1\bar{B}_0$$

$$F = \bar{\bar{F}}$$

Example:- Simplify $F(A,B,C) = \underbrace{\Pi_M(0,1,5)}_{0's} + \underbrace{\sum_d(4,6)}_{\text{Don't care}}$

as SOP



$F = B$ / $\bar{F} = \bar{B}$

Five Apple

12/7/2016

2.5 other types of gates :-

- Buffer
- Tri-state Buffer
- NAND
- NOR
- XOR
- XNOR

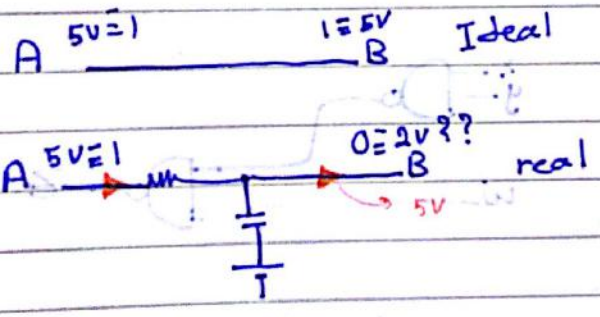
* Buffer

x	y
0	0
1	1

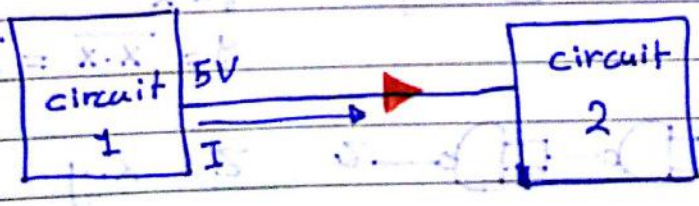
$y = x$

It's used for two perposes :-

① signal amplification ...



② Isolation and protection :-

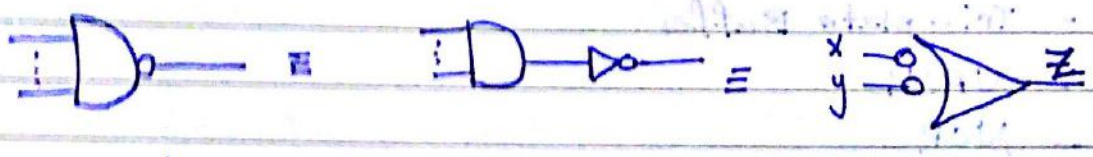


Five Apple

12/7/2016

* NAND gate

* NAND = NOT-AND

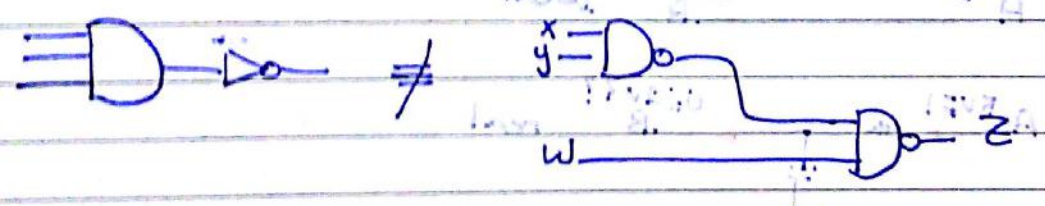


x	y	z
0	0	1
0	1	1
1	0	1
1	1	0

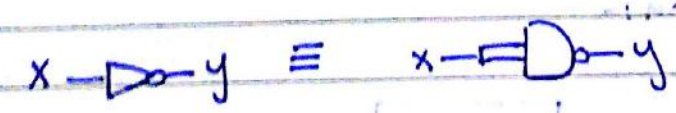
عكس and

$z = \overline{x \cdot y} = \overline{x} + \overline{y}$

3-input NAND

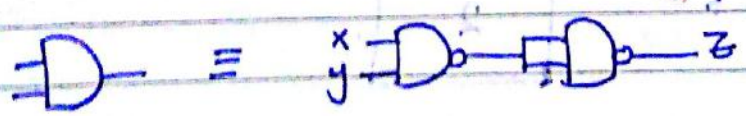


⇒ The NAND gate is a universal gate



$y = \overline{x}$

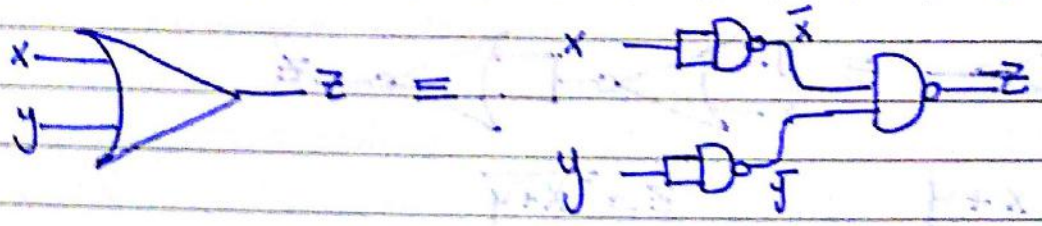
$y = \overline{x \cdot x} = \overline{x}$



$z = x \cdot y$

$z = \overline{\overline{x \cdot y}} = x \cdot y$

12/7/2016



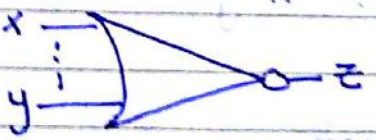
$z = x + y$

$z = \overline{\bar{x} \cdot \bar{y}}$

$= \bar{\bar{x}} + \bar{\bar{y}} = x + y$

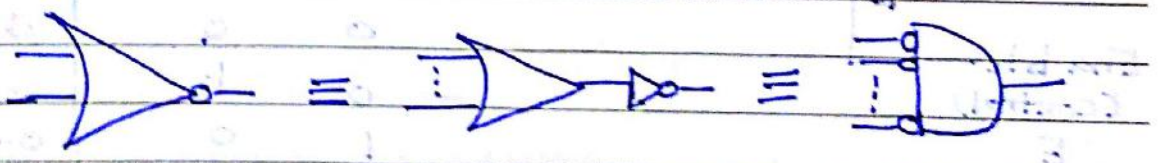
* NOR gate ≡ NOT-OR

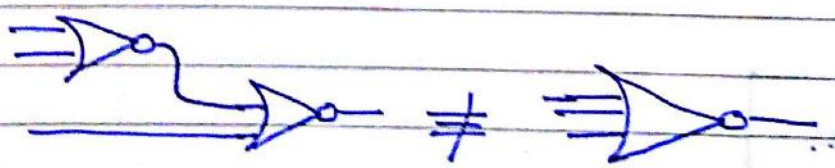
OR-NOT gate →



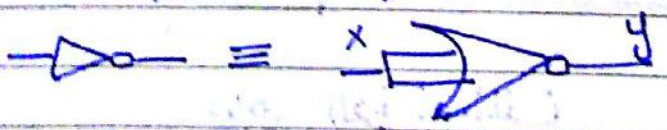
x	y	
0	0	1
0	1	0
1	0	0
1	1	0

$z = \overline{x+y} = \bar{x} \cdot \bar{y}$





* The NOR gate is a universal gate

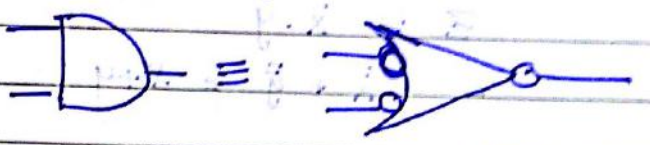


$y = \bar{x}$

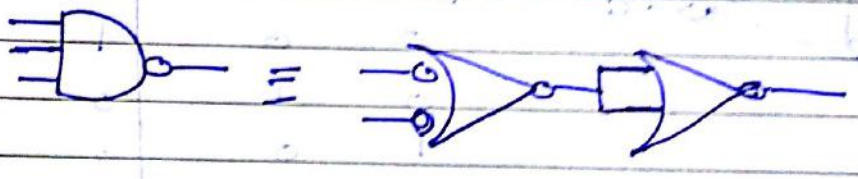
$y = \overline{x + x} = \bar{x}$

12/7/2016

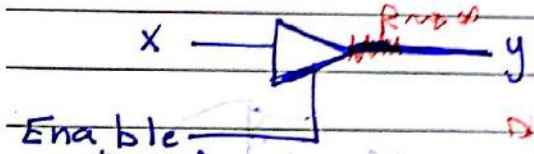
$z = x + y$ $\quad$ $z = \overline{x+y}$



$x \cdot y = z$ $\quad$ $z = \overline{\bar{x} + \bar{y}} = \bar{\bar{x}} \cdot \bar{y} = x \cdot y$



Tri-state Buffer :-

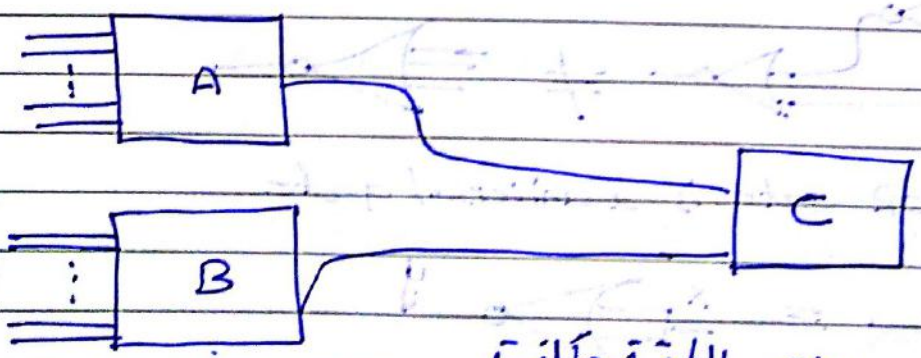


output is electrically disconnected

High impedance (Resistance) internally

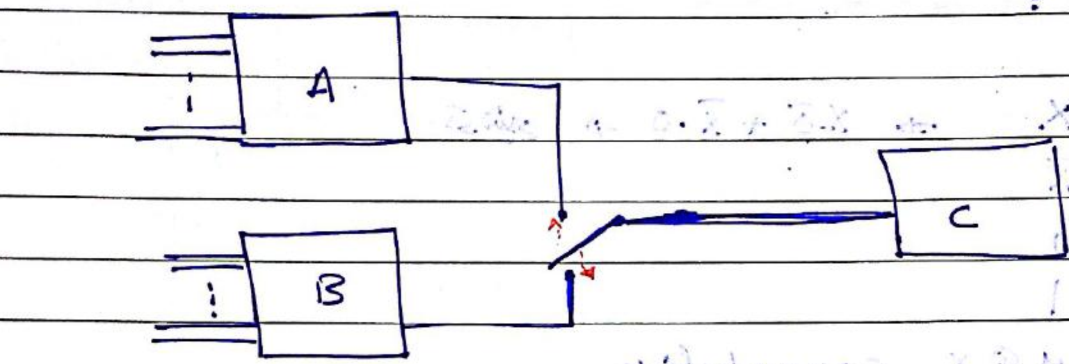
E	x	y
Disabled: 0	0	Z
Disabled: 0	1	Z
Enabled: 1	0	0
Enabled: 1	1	1

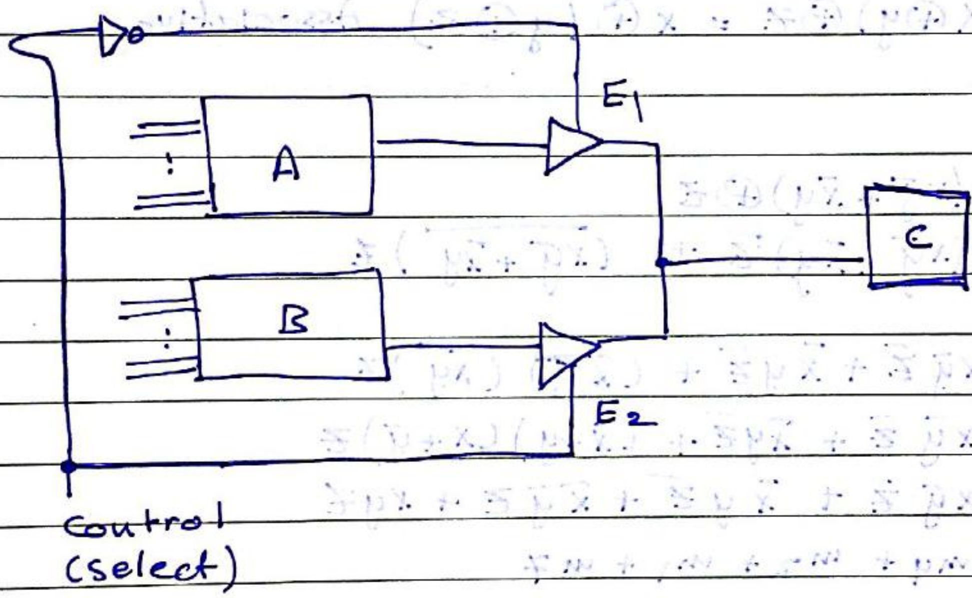
$\rightarrow y = x$



وهي الطريقة.. مكلفة

12/7/2016





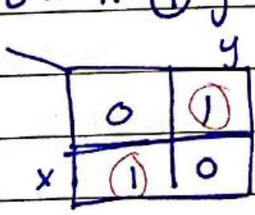
* XOR gate

⇒ Exclusive - OR

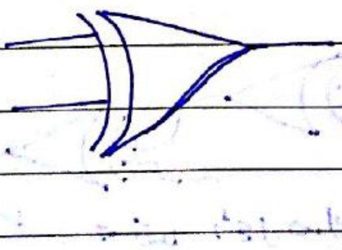
X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	0

لحتى يطلع الجواب (1)
لازم بس (1) يكون
1 و الثاني 0
وما يصير التنين
تأخدوا (1)

$Z = X \oplus y$



$Z = X \oplus y = X\cdot\bar{y} + \bar{X}y$



← شكل ال XOR

12/7/2016

* Properties :-

$x \oplus 0 = X$ ⇒ $X.\bar{0} + \bar{X}.0$ ⇒ تبديلة

$x \oplus 1 = \bar{X}$

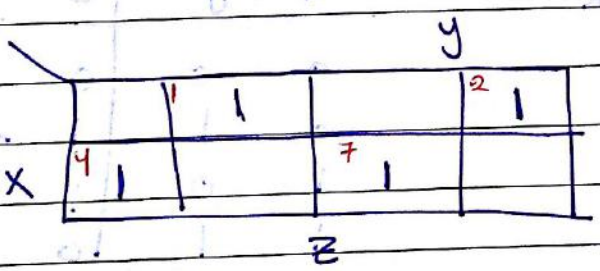
$x \oplus x = 0$

$x \oplus \bar{x} = 1$

$x \oplus y = y \oplus x$ commutative

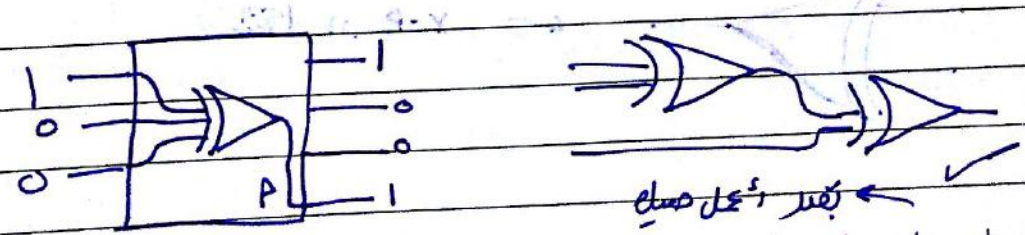
$x \oplus y \oplus z = (x \oplus y) \oplus z = x \oplus (y \oplus z)$ Associative

$$x \oplus y \oplus z = (x\bar{y} + \bar{x}y) \oplus z$$
$$= (x\bar{y} + \bar{x}y)\bar{z} + \overline{(x\bar{y} + \bar{x}y)}\,z$$
$$= x\bar{y}\bar{z} + \bar{x}y\bar{z} + (\bar{x}\bar{y})(\bar{x}y)z$$
$$= x\bar{y}\bar{z} + \bar{x}y\bar{z} + (\bar{x}+y)(x+\bar{y})z$$
$$= x\bar{y}\bar{z} + \bar{x}y\bar{z} + \bar{x}\bar{y}z + xyz$$
$$= m_4 + m_2 + m_1 + m_7$$
$$= \sum_m (1, 2, 4, 7)$$

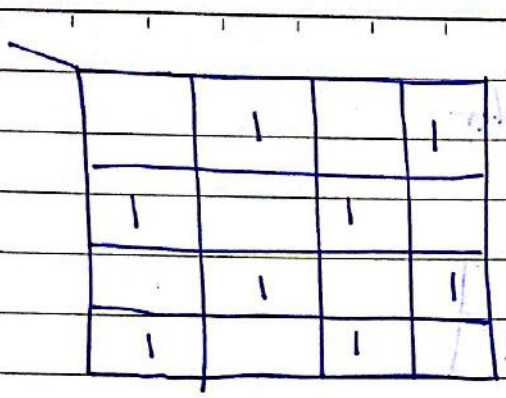


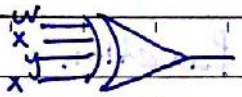
⇒ XOR ⇒ odd function

⇒ Even parity generator



12/7/2016





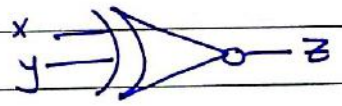
$w \oplus y \oplus x \oplus z$

* XNOR gate

Exclusive-not-OR

x	y	
0	0	1
0	1	0
1	0	0
1	1	1

← NOR



2-input equal

$$z = x \odot y = \overline{x \oplus y} = \overline{x\bar{y} + \bar{x}y}$$
$$= xy + \bar{x}\bar{y}$$

properties

$x \odot 0 = \bar{x}$

$x \odot 1 = x$

$x \odot x = 1$

$x \odot \bar{x} = 0$

$x \odot y = y \odot x$

not Associative

12/7/2016

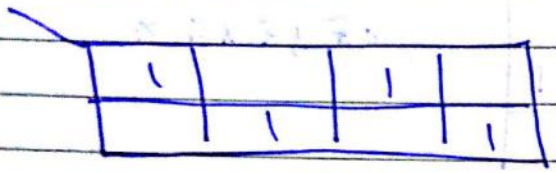
XNOR ⇒ even function

⇒ odd parity generator

$X \odot Y \odot Z$

12/7/2016

XNOR ⇒ even function
⇒ odd parity generator



$X \odot y \odot z$

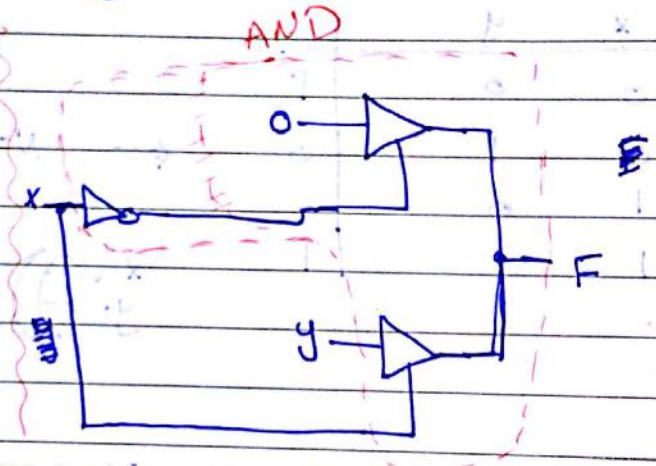
Example 1- Implement a 2-input AND gate using: Tri-state buffers and inverters only.

control

	x	y	
x=0	0	0	0
	0	1	0
x=1	1	0	0
	1	1	1

F=0

F=y



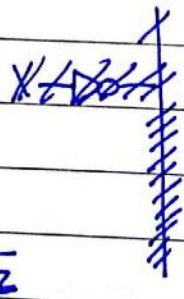
control

Ex:-

	x	y	z	F
x=0	0	0	0	0
	0	0	1	1
	0	1	0	1
	0	1	1	1
x=1	1	0	0	1
	1	0	1	0
	1	1	0	0
	1	1	1	0

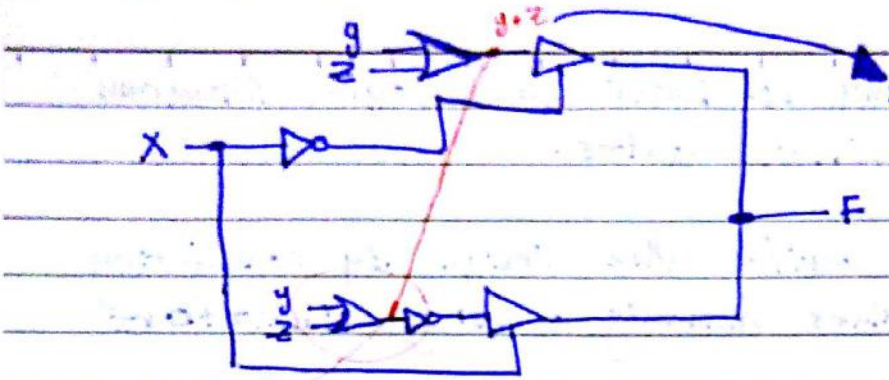
using (tri-state buffer & inverters/ OR)

F = y + z



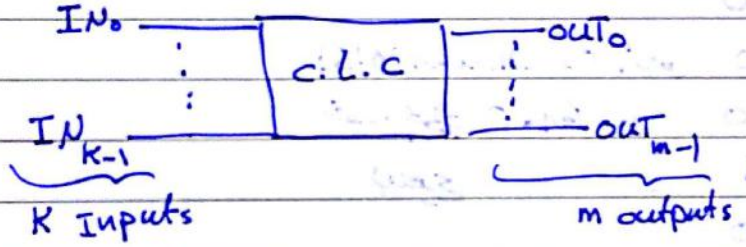
$f = \overline{y+z}$

13/7/2016



CH. 3 Combinational Logic Design :-

3.1 Design procedure



$$OUT_i = f(I_{N0}, IN_1, \dots, IN_{K-1})$$

* output depends only on the Input value...

Design steps :-

1. specification ⇒ speeify and describe the operation of the circuit.
2. formulation ⇒ derive an initial logic expression or the T.T
3. Optimization ⇒ Determin the semplified SOP or POS expression using K-maps

13/7/2016

4- Technology mapping ⇒ Draw the Logic Diagram using AOI, NOR, NAND gates.

5- verification ⇒ verify the design By considering different input values manually or using simulators

Example 1- Design a 3-input majority circuit. → specification

⇒ the number of Inputs = 3 X_2, X_1, X_0

\# outputs = 1

F = 1 when the Input has more 1s than 0s.

(2)

X_2	X_1	X_0	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

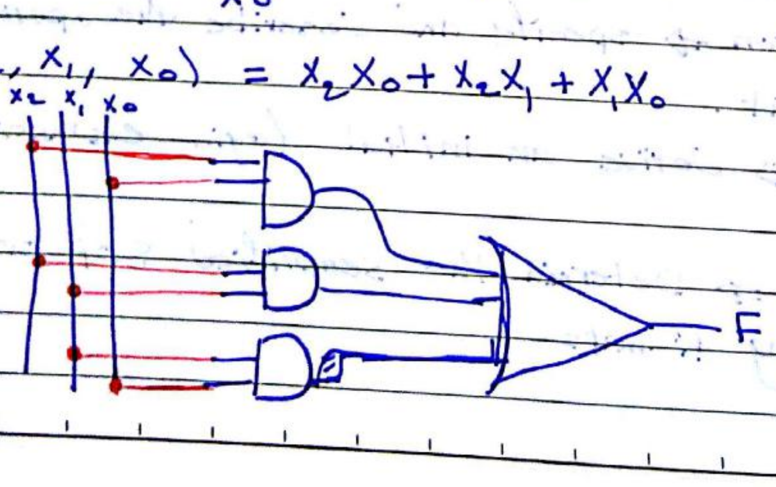
إذا كان عدد ال 0s أكثر
عند الجواب 0 والعكس
صحيح

(3) F

x_2 \ $x_1 x_0$			X_1	
	.	.	1	.
X_2	.	1	1	1
		X_0		

$F(X_2, X_1, X_0) = X_2X_0 + X_2X_1 + X_1X_0$

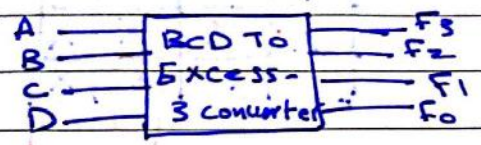
(4)



13/7/2016

⑤ verify the Design ...

Example :- Design a c/c that convert a 4-digit BCD number into Excess 3 code ⇒ BCD + 3



Input = 4 # output = 4

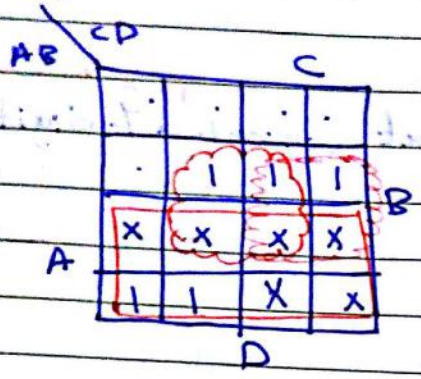
{F3, F2, F1, F0} = {A, B, C, D} + 3

②

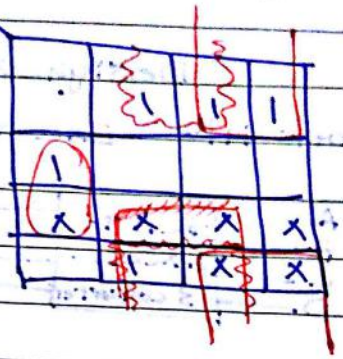
A	B	C	D	F3	F2	F1	F0
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0
1	0	1	0	X	X	X	X
1	0	1	1	X	X	X	X
1	1	0	0	X	X	X	X
1	1	0	1	X	X	X	X
1	1	1	0	X	X	X	X
1	1	1	1	X	X	X	X

13/7/2016

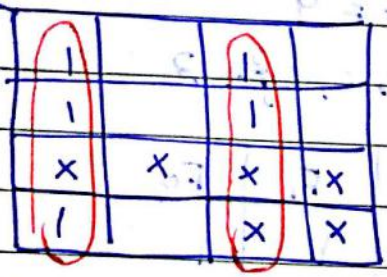
③ F_3 → $F_3(A,B,C,D) = A + \bar{B}D + \bar{B}C$



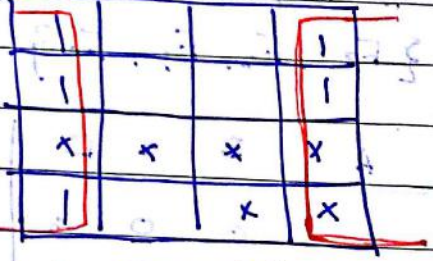
F_2 $F_2(A,B,C,D) = B\bar{C}\bar{D} + \bar{B}D + \bar{B}C$



F_1



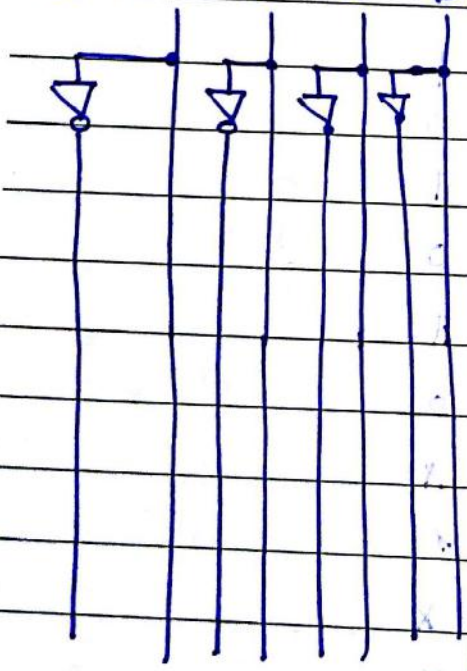
F_0

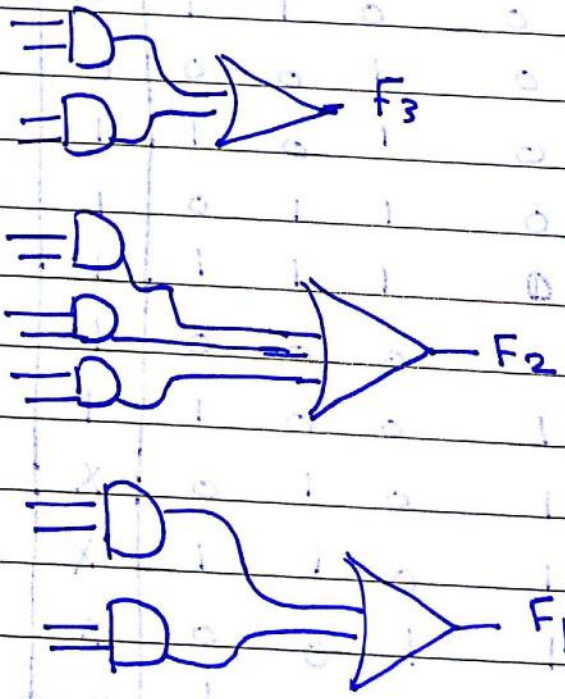


$F_1(A,B,C,D) = \bar{C}\bar{D} + CD = C \odot D$

$F_0 = \bar{D}$

④ A B C D





8/7/2016

Example :- Design a circuit that compares two 2-bit binary number A & B and outputs two signals; O_1 & O_0 such that :-

O_1O_0 = { 00 ; A == B and both are even
01 ; A < B
10 ; A > B
11 ; A == B and both are odd

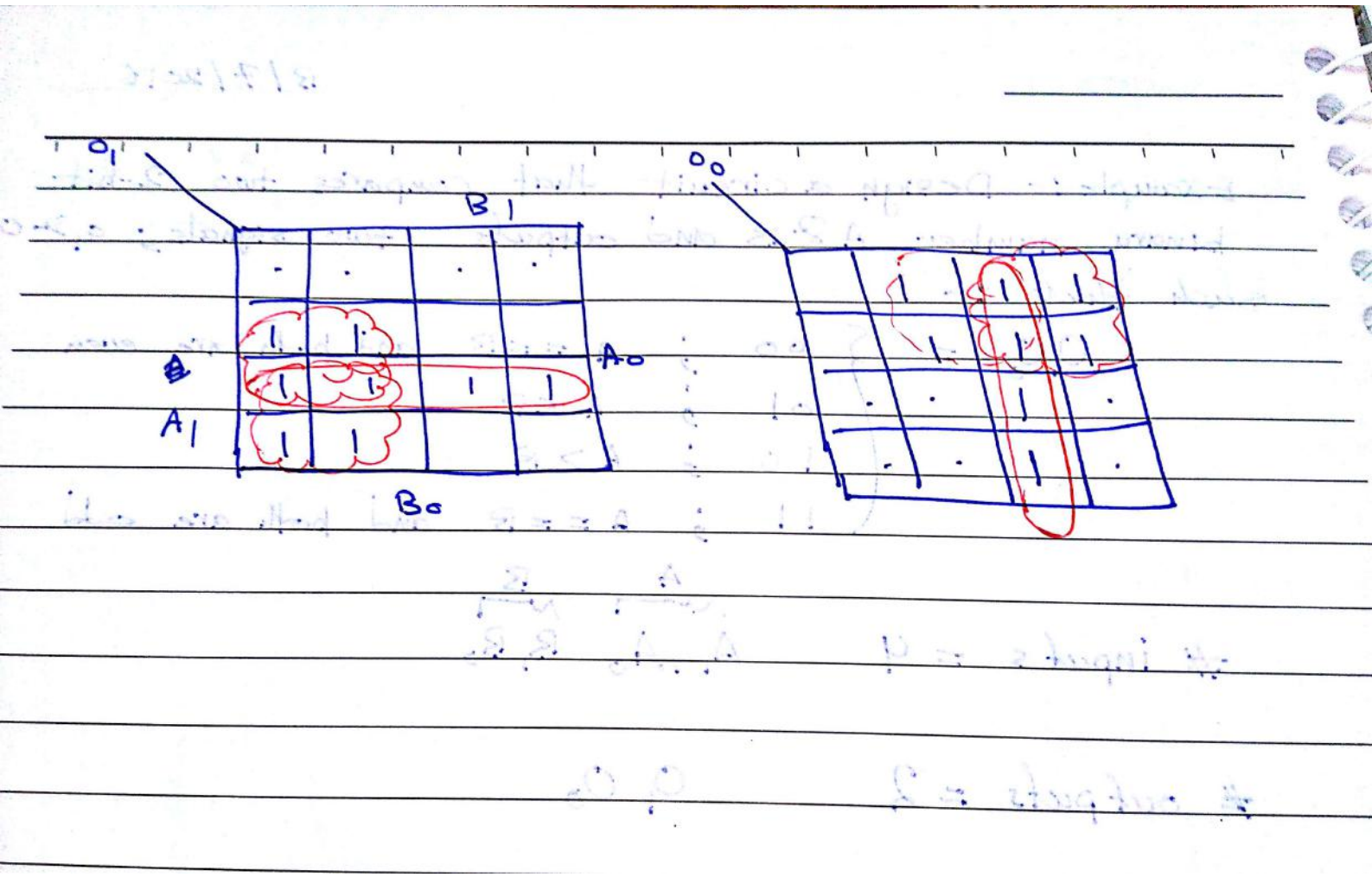
A B

inputs = 4 A_1, A_0 B_1, B_0

outputs = 2 O_1 O_0

A_1A_0	B_1	B_0	O_1	O_0
0	0	0	0	0
0	0	1	0	1
0	0	1	0	1
0	0	1	0	1
0	1	0	1	0
0	1	0	1	1
0	1	1	0	1
0	1	1	0	1
1	0	0	1	0
1	0	0	1	0
1	0	1	0	0
1	0	1	0	1
1	0	0	1	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1

Five Apple



17/7/2016

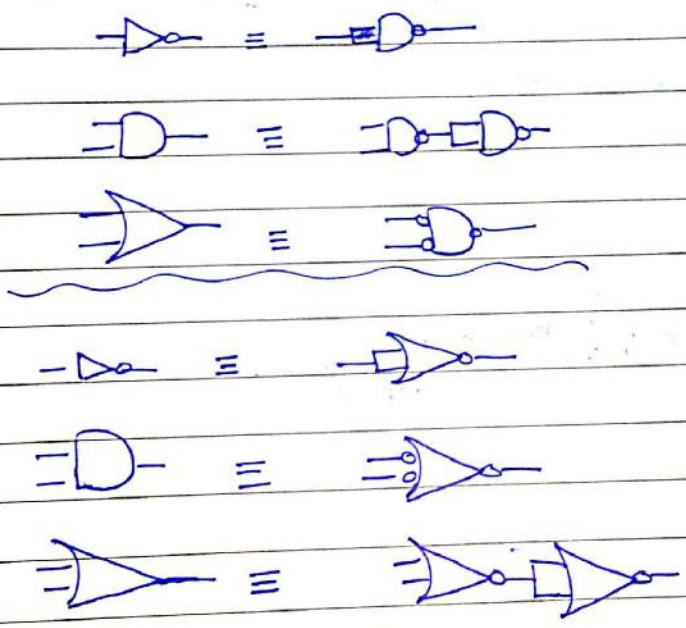
3.2 Technology Maping :-

How to realize/implement a logic circuit using NAND or NOR gates only ?!

* How to convert a logic circuit implemented using AOI to ~~an~~ NAND or NOR implementation ?!

Steps:- given an AOI Logic Diagram

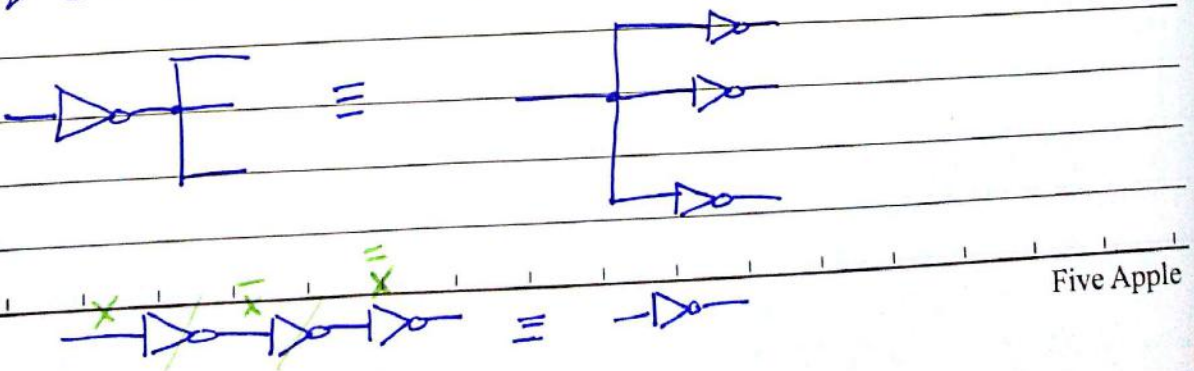
① Replace AOI gates by their NAND or NOR equivalent



② Repeat the following ~~until~~ until there in are inverter at most in each branch.

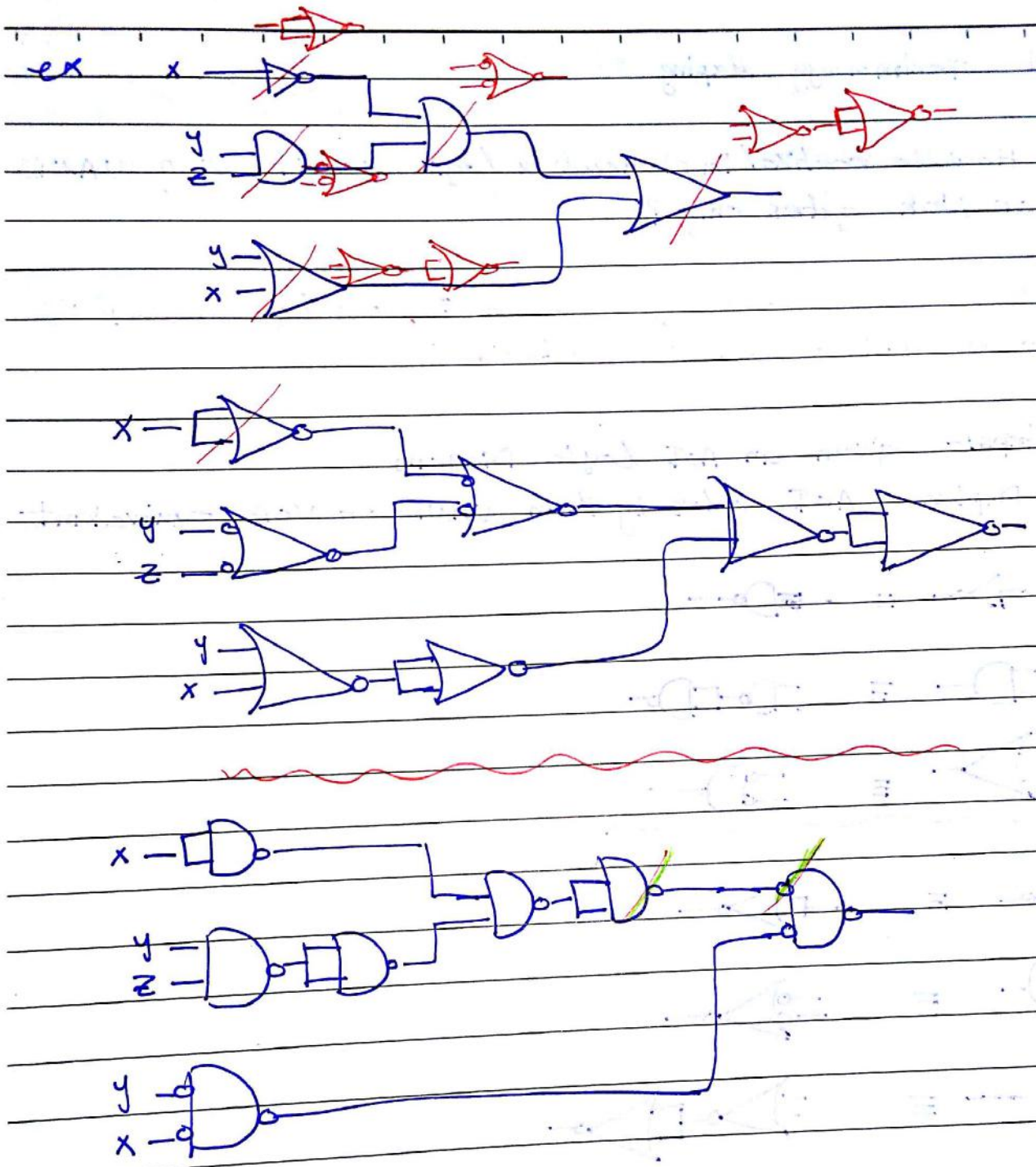
→ push inverters through fan~~point~~-out points.

→ cancel inverter pairs



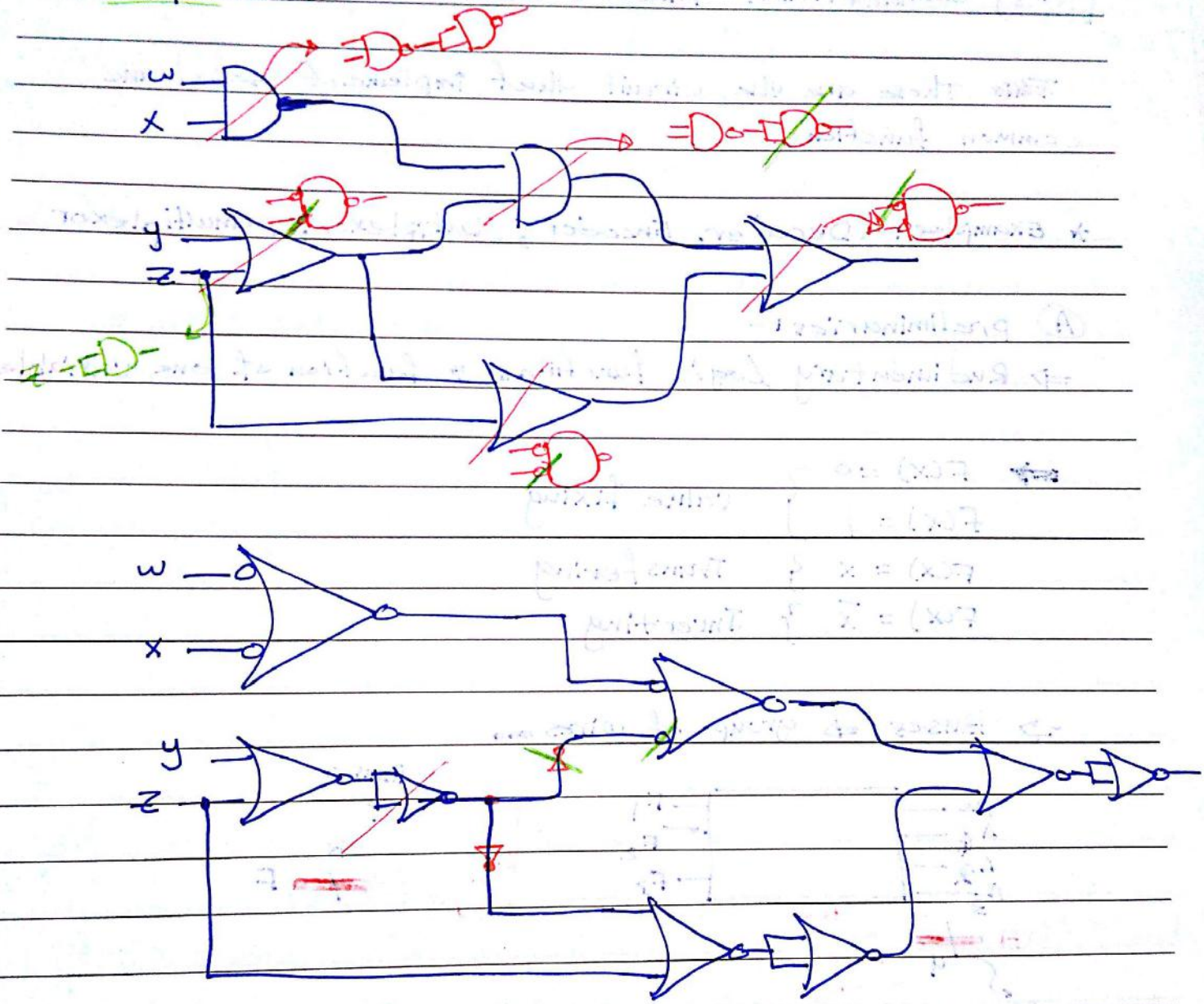
Five Apple

17/7/2016



17/7/2016

Example 8-



17/7/2016

3.3 Combinational Function Blocks :-

These are the circuit that implement useful and common functions.

* Examples :- Decoder, Encoder, Multiplexor, Demultiplexor

(A) preliminaries :-

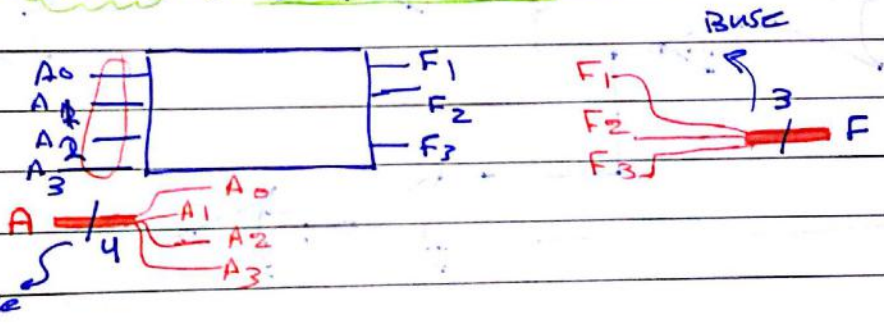
⇒ Rudimentary Logic functions → function of one variable

➡ $F(x) = 0$, $F(x) = 1$ } value fixing

$F(x) = x$ } Transfering

$F(x) = \bar{x}$ } Inverting

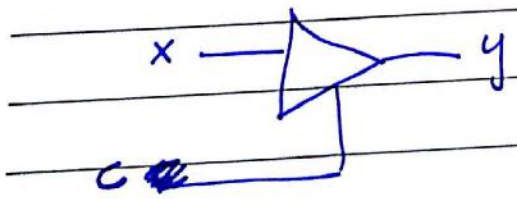
⇒ Buses ⇒ group of wires



* Enabling circuits :-

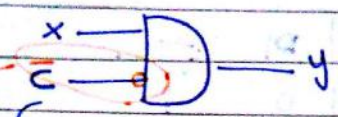
⇒ circuits that allow the input to pass through to the output

⇒ using Tri-state buffer



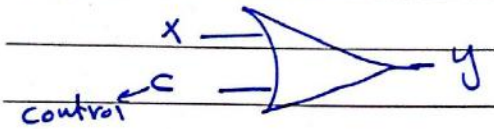
17/7/2016

→ using and gate



$$y = \begin{cases} 0, & c = 0 \\ x, & c = 1 \end{cases}$$ ← نفس النتائج x

→ using OR gate



$$y = \begin{cases} x, & c = 0 \\ 1, & c = 1 \end{cases}$$

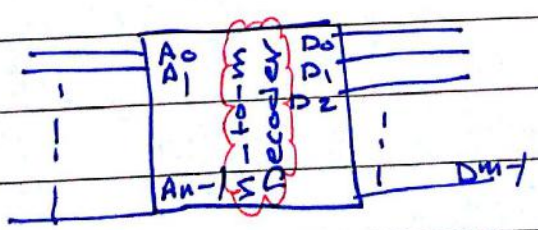
(B) Decoder :- (line decoders)

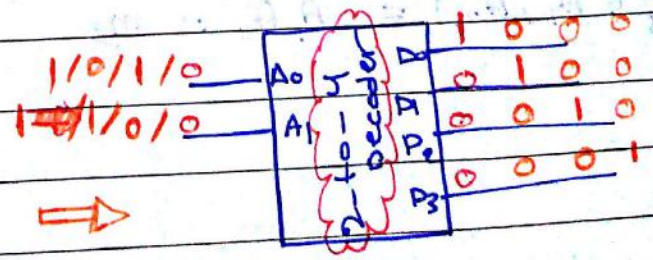
* A decoder is a circuit that converts an n-bit input (code) to m-bit output such that :-

(a) the relation $n \leq m \leq 2^n$ is satisfied...

(b) only one of the m outputs is active for any possible input (0 or 1)

(c) The active output is the one whose number / table is the decimal value of the binary input.





Five Appl

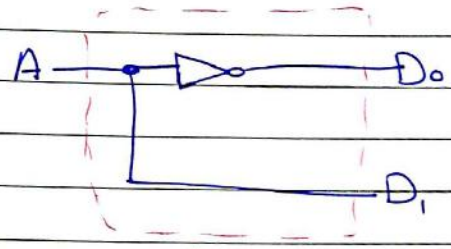
17/7/2016

Design of a 1-to-2 decoder

A	D_1	D_0
0	0	1
1	1	0

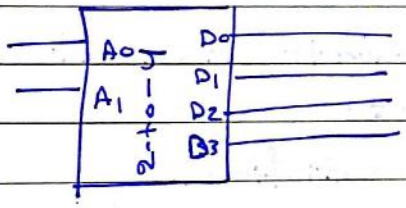
$D_0(A) = \bar{A}$

$D_1(A) = A$



1-to-2

Design of 2-to-4 decoder.



A_1	A_0	D_3	D_2	D_1	D_0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

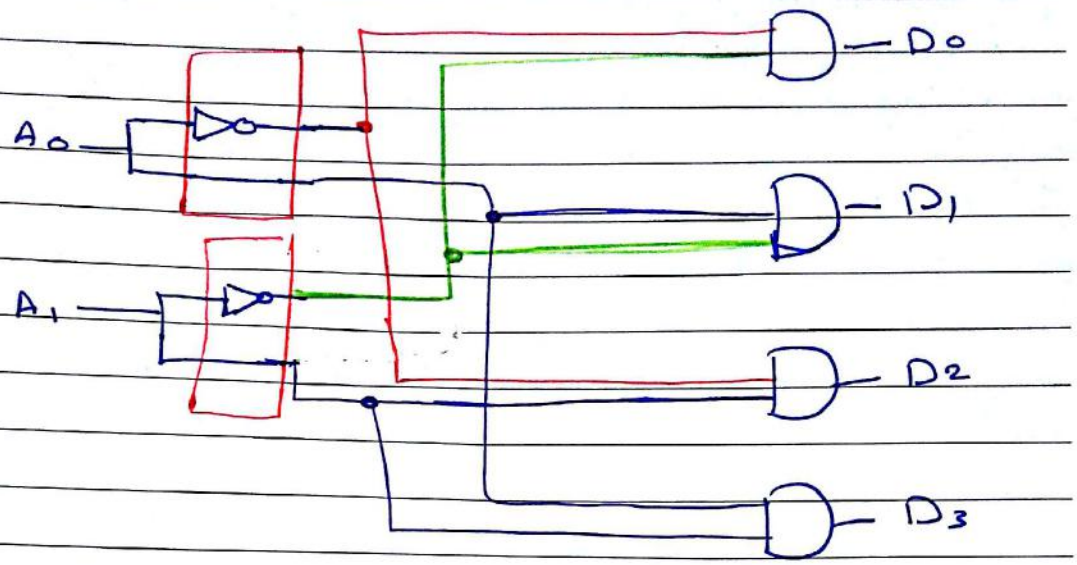
$D_0(A_1, A_2) = \bar{A}_1 \bar{A}_0 = m_0$

$D_1(A_1, A_2) = \bar{A}_1 A_0 = m_1$

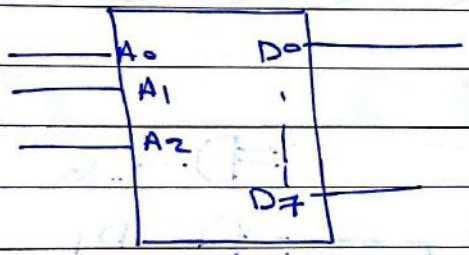
$D_2(A_1, A_2) = A_1 \bar{A}_0 = m_2$

$D_3(A_1, A_2) = A_1 A_0 = m_3$

17/7/2016



* Design 3-to-8 decoder



4-to-16 : 4 to 2 , 16 4input AND

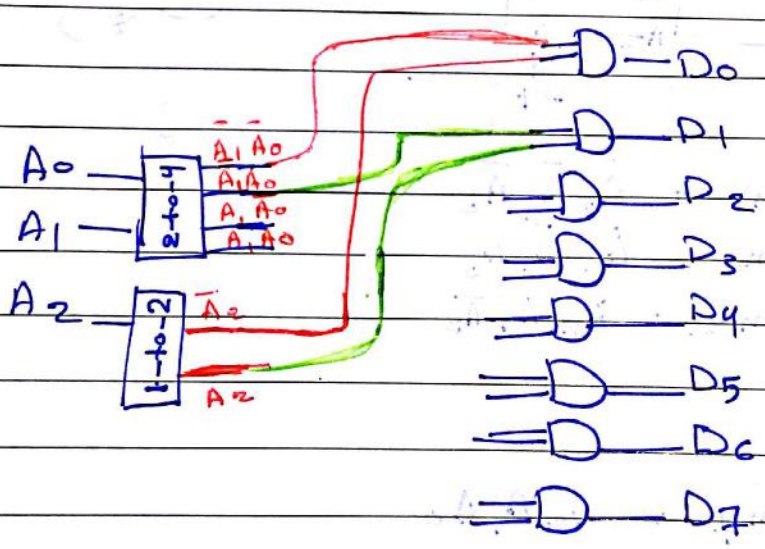
$D_0 = m_0 = \bar{A}_2 \bar{A}_1 \bar{A}_0$

$D_1 = m_1 = \bar{A}_2 \bar{A}_1 A_0$

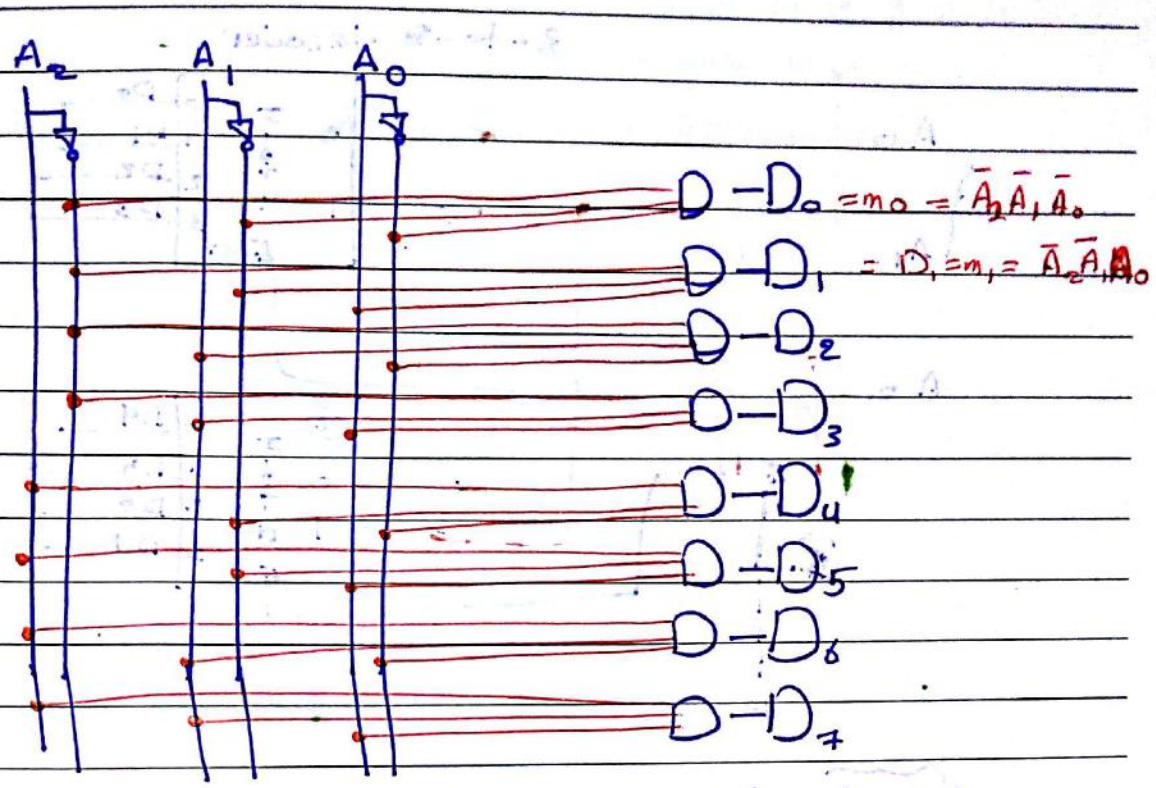
$D_2 = m_2 = \bar{A}_2 A_1 \bar{A}_0$

!

$D_7 = m_7 = A_2 A_1 A_0$



18/7/2016

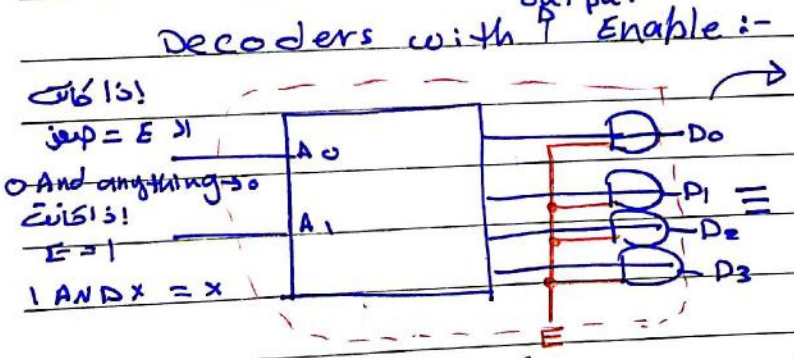


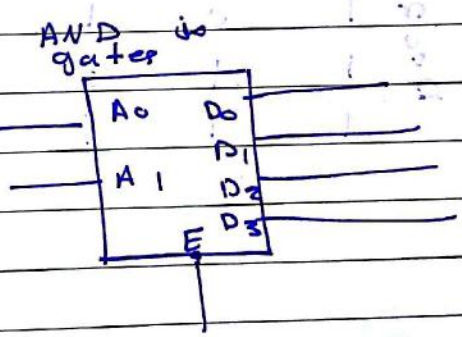
n-to-m decoder

n → 1-to-2 decoder

m → n-input AND gates

Decoders with output Enable :-



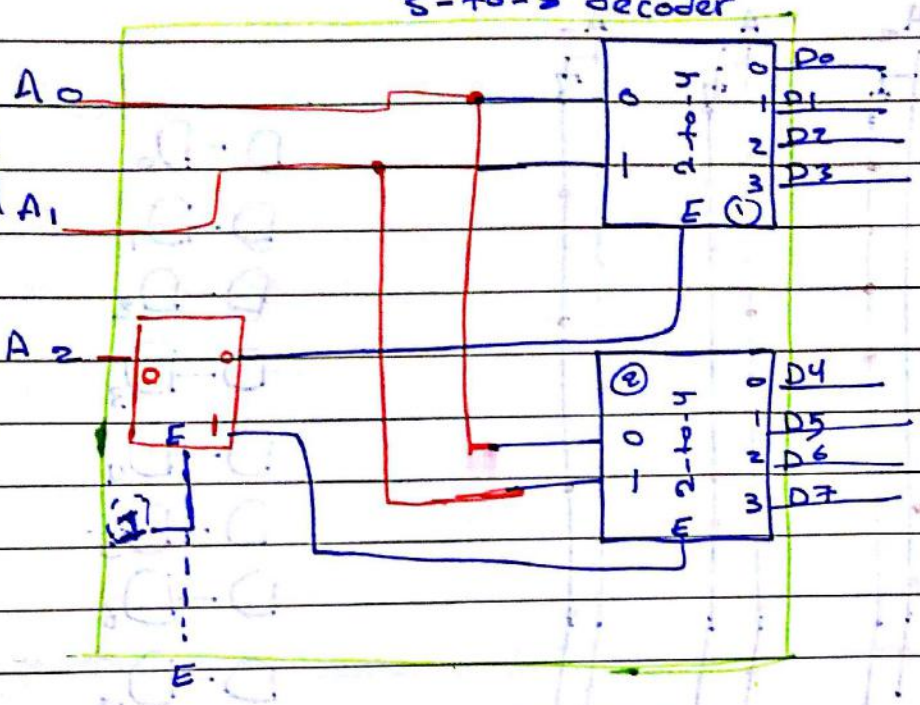


* Can be used to build bigger decoders

Example :- assume that you have as many 1-to-2 and 2-to-4 decoders with enable, Build 3-to-8 decoders

18/7/2016

3-to-8 decoder

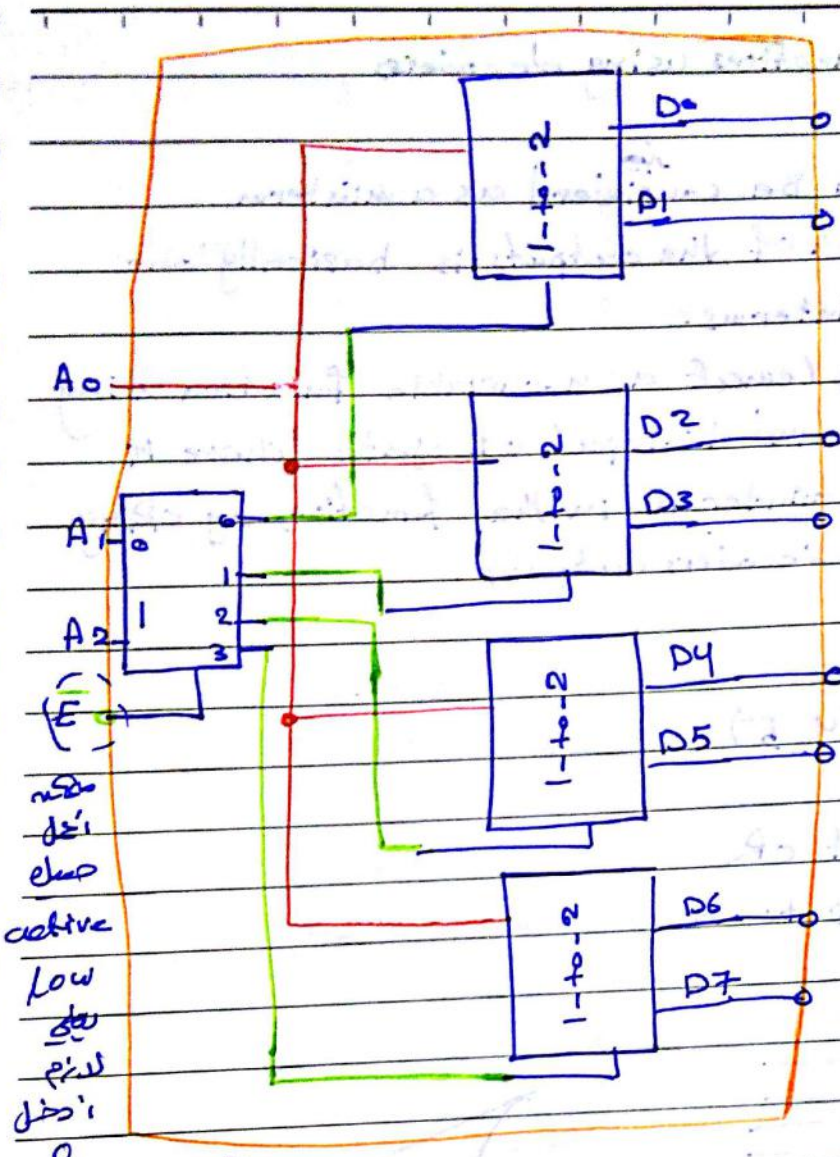


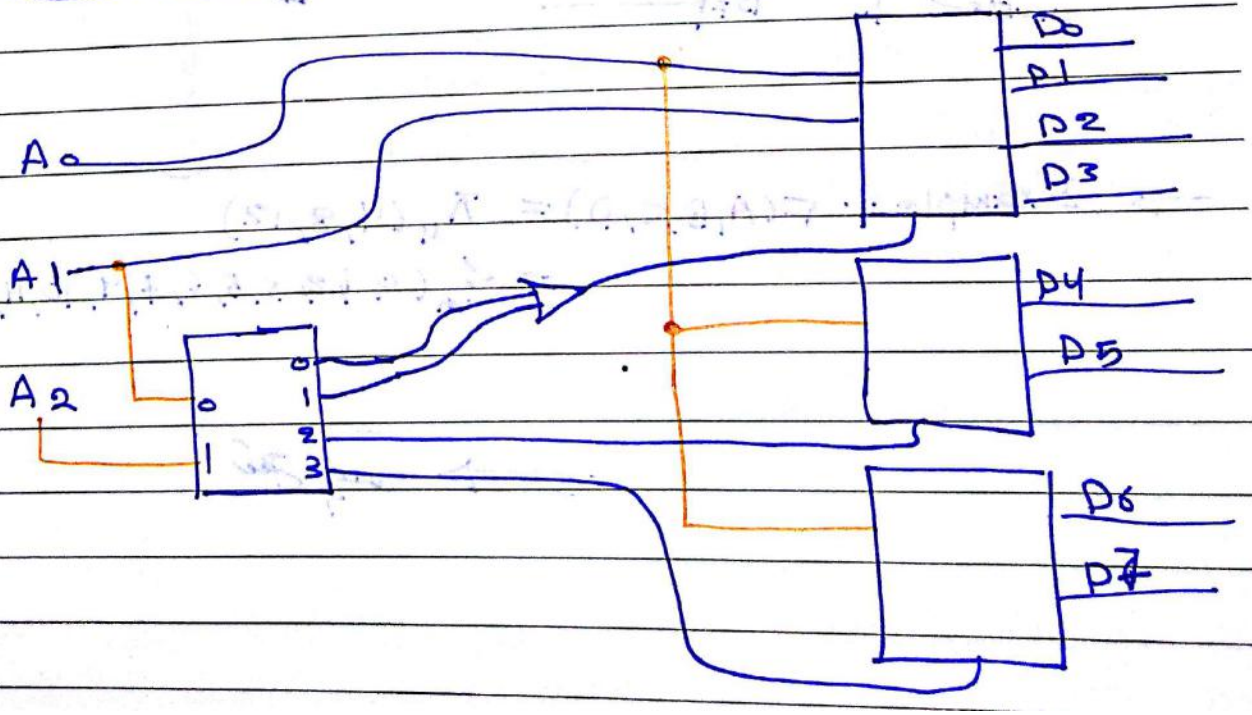
A_2	A_1	A_0	
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

لما بيكون الـ E ← صفر معناته D_2 مطفي بس
D_1 بيشتغل.

D_1 مطفي بس D_2 بيشتغل وهكذا !!

18/7/2016





13/7/2016

Implementing logic functions using decoders

⇒ The decoder can be considered as a minterm generator, i.e each of the outputs is basically one of the possible minterms.

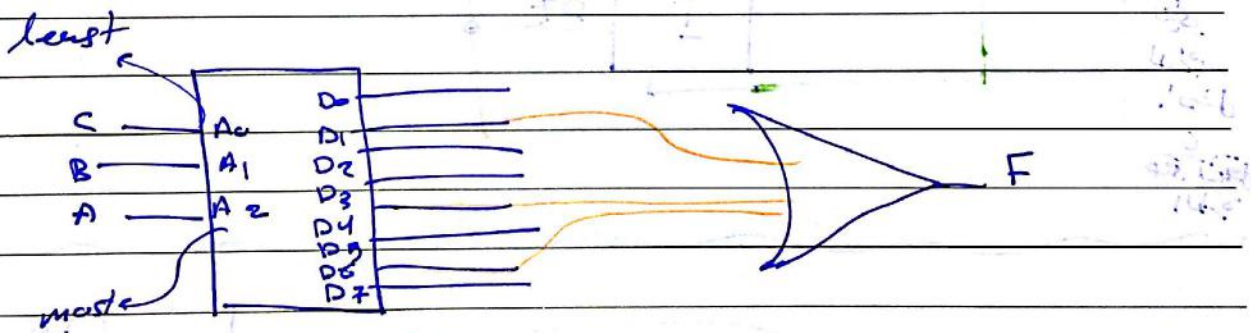
⇒ Thus, we can implement a n-variable function using n-to-m decoder and k-input of gate where k is the number of minterms in the function, by oRing the corresponding decoders outputs.

Example 1-

$F(A,B,C) = \sum_m (1,4,5)$

K = 3 ⇒ 3 input OR

3 variables ⇒ 3-to-8

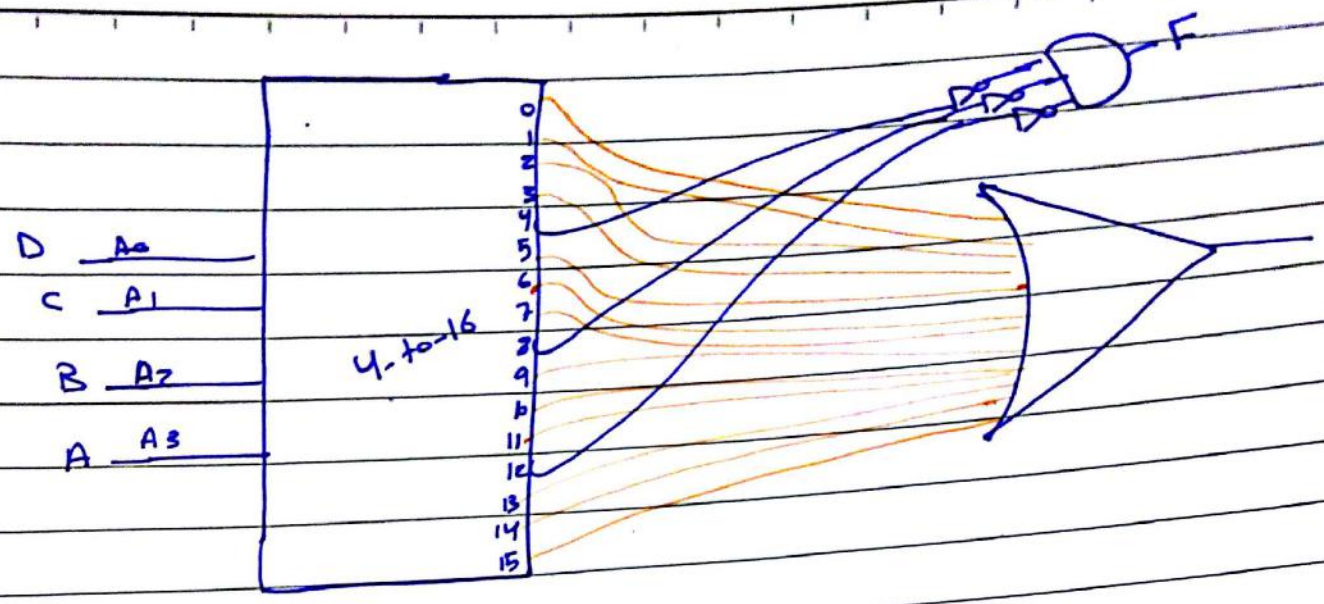


⇒ Example :- $F(A,B,C,D) = \prod_M (4,8,12)$

$= \sum_m (0,1,2,3,5,6,7,9,10,11,13,14,15)$

⇒ [illegible]

12/7/2016



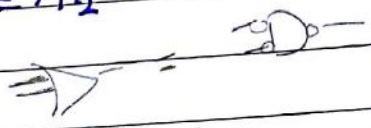
4 variables ⟹ 4-to-16 decoder

13 minterms ⟹ 13-input-OR

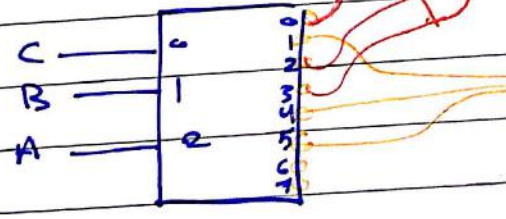
Note :- $m_0 = \bar{A}\bar{B}\bar{C}\bar{D}$ ↖ $m_0 = \bar{M}_0$

$M_0 = A + B + C + D$ ↗

$m_2 = \bar{M}_2$ OR $\bar{m}_2 = M_2$



Example :-



$F(A,B,C) = \sum_m^1 (0,2,3,6,7)$

$F = \pi_M (1,4,5)$

$F(A,B,C) = \sum_m (0,2,3)$

$= \pi_M (1,4,5,6,7)$

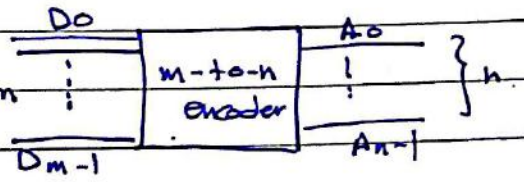
19/7/2016

(B) Encoder

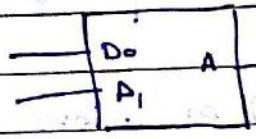
* is a combinational Logic circuit with m inputs and n outputs such that :-

a) The relation $m \geq n \geq \log_2 m$ ← Decoder ليس كذلك holds.

b) the output value is basically the bit position of the single '1' is the input.

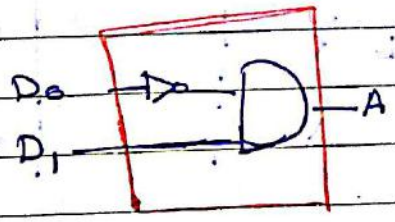


* 2-to-1 encoder ⟹



D_1	D_0	A
?	0	0 (x will never occure)
0	1	0
1	0	1
?	1	0 (x)

$A = m_2 = D_1 \overline{D_0}$



* Assuming the output to 'x' for invalid input & defining an additional output V to indicate whether the input is valid or not

D_1	D_0	A	V
0	0	x	0
0	1	0	1
1	0	1	1
1	1	x	0

A

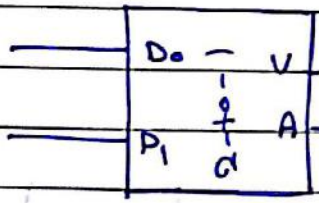
	D_0	
	X	0
D_1	1	X

$A = D_1 = \bar{D}_0$

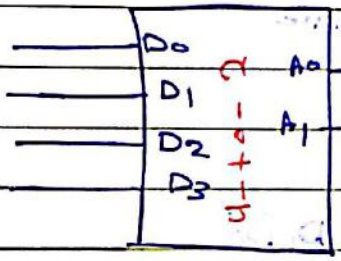
V

	D_0	
	0	1
D_1	1	0

$V = D_1 \oplus D_0$



* 4-to-2 encoder



D_3	D_2	D_1	D_0	A_1	A_0	V
0	0	0	1	0	0	1
0	0	1	0	0	1	1
0	1	0	0	1	0	1
1	0	0	0	1	1	1
ما بهمني ^^				0	0	
				0	0	
				0	0	

$\Rightarrow A_1 = m_4 + m_8 = \bar{D}_3 D_2 \bar{D}_1 \bar{D}_0 + D_3 \bar{D}_2 \bar{D}_1 \bar{D}_0$

$A_0 = m_2 + m_3 = \bar{D}_3 \bar{D}_2 D_1 \bar{D}_0 + D_3 \bar{D}_2 \bar{D}_1 \bar{D}_0$

OR from T.T directly

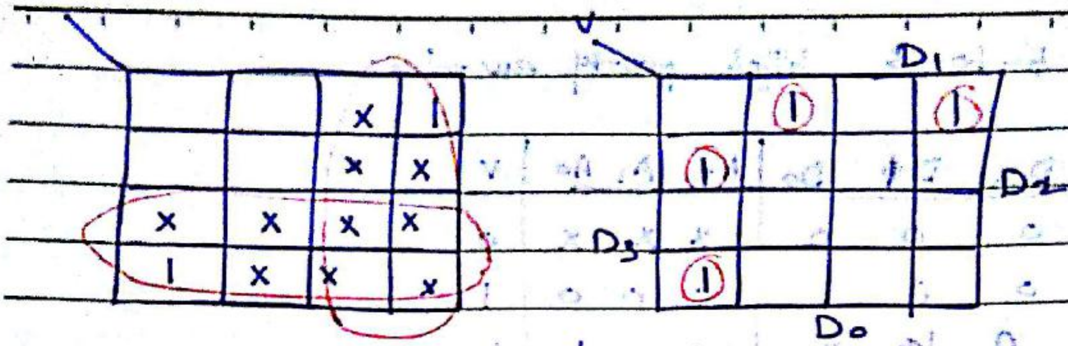
$A_1 = D_2 + D_3$

$A_0 = D_1 + D_3$

A_1

D_3 \ D_2				
	1	X	X	X
	X	X	X	X
	1	X	X	X

19/7/2016



* priority encoders :-

They can deal with Input combination with more than 1. The output is basically the highest or the lowest bit position having a 1

highest → high priority encoder

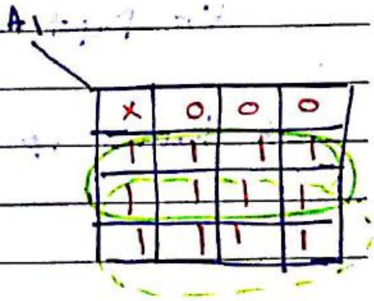
lowest → low priority encoder

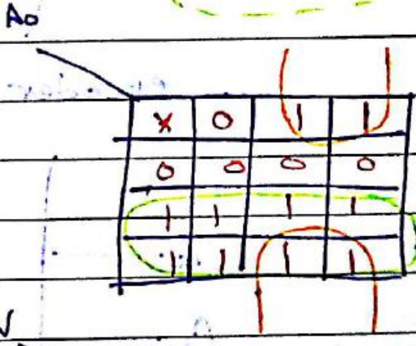
* 4-to-2 high priority encoder :-

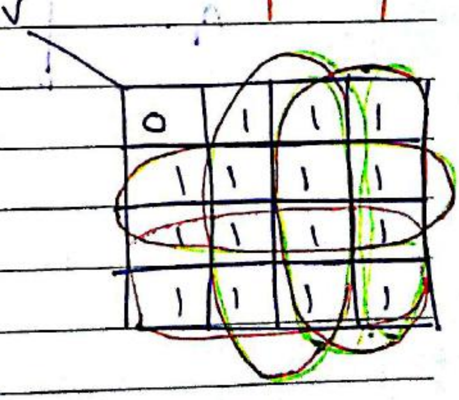
D3	D2	D1	D0	A1	A0	V
0	0	0	0	x	x	0
0	0	0	1	0	0	1
0	0	1	X (0)	0	1	1
0	0	1	0	0	1	1
0	1	x	x	1	0	1
1	x	x	x	1	1	1

نفس كأنه دمجت عليه بسطر واحد

برمج سطور لأنه ما بفرق معه شو ما موجود ...







→ $A_1 = D_3 + D_2$

→ $A_0 = D_3 + \overline{D_2} D_1$

→ $V = D_3 + D_2 + D_1 + D_0$

19/7/2016

Example :- 5-to-3 high prioty encoder

D_4	D_3	D_2	$D\phi$	D_0	A_2	A_1	A_0	V
0	0	0	0	0	x	x	x	0
0	0	0	0	1	0	0	0	1
0	0	0	1	X	0	0	ϕ	1
0	0	1	X	X	0	1	0	1
0	1	X	X	X	0	1	1	1
1	X	X	X	X	1	0	0	1

$A_2 = D_4$

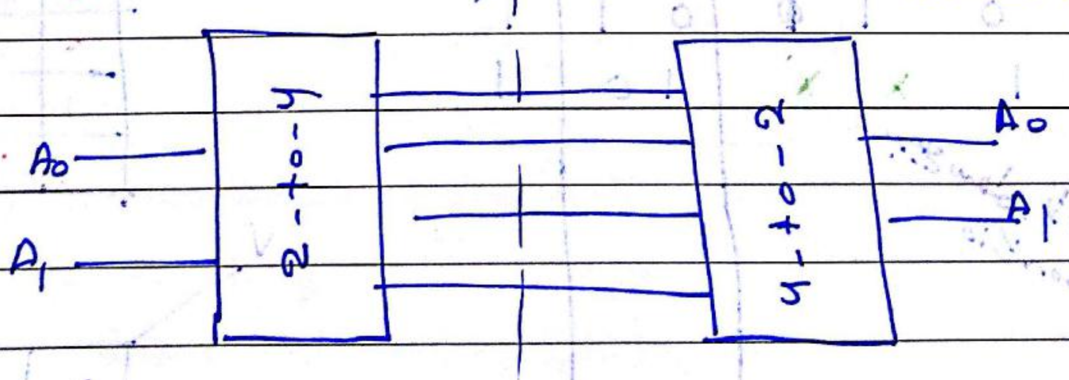
$A_1 =$ ~~[struck through]~~ $A_1 = \overline{D_4} D_3 + \overline{D_4}\, \overline{D_3} D_2$

$A_0 = D_3 \overline{D_4} + \overline{D_4}\, \overline{D_3}\, \overline{D_2} D_1$

$V = D_4 + D_3 + D_2 + D_1 + D_0$

at home ⇒ { 6-to-3 high
6-to-3 low

* ملاحظة العلاقة اللي بتحكم بال Decoder وال encoder ...



أخذ العكس

20/7/2016

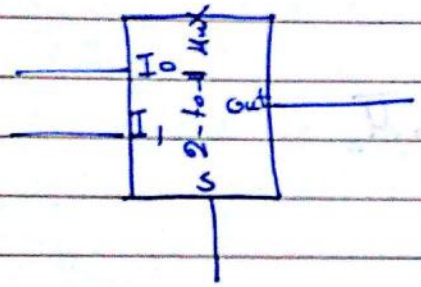
⑥ Multiplexor :- (Data selector)

* It's a combinational logic circuit with (m) inputs and a single output, such that :

(a) one of the inputs is selected to appear at the output ...

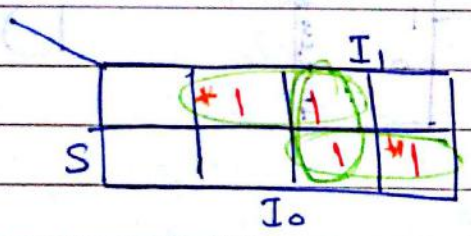
(b) the selected input is specified using ... an additional set of $\lceil \log_2 m \rceil$ inputs. These inputs are called the select lines ...

* 2-to-1 MuX

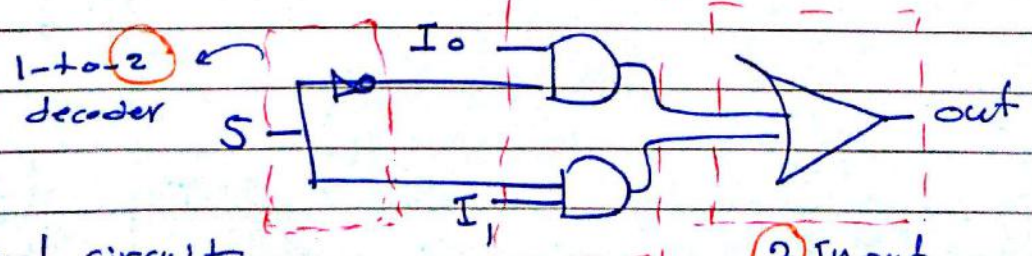


S	I_1	I_0	out
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

$$OUT = \begin{cases} I_0, & S=0 \\ I_1, & S=1 \end{cases}$$



$$OUT(S, I_1, I_0) = \bar{S} I_0 + S I_1$$ — (2)-to-1 MuX



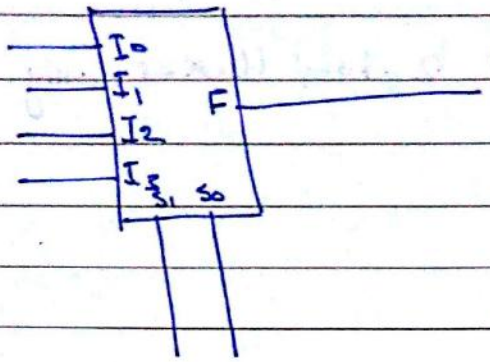
3-level circuit ...

(2) Input OR gate

2 (2)-Input and gate

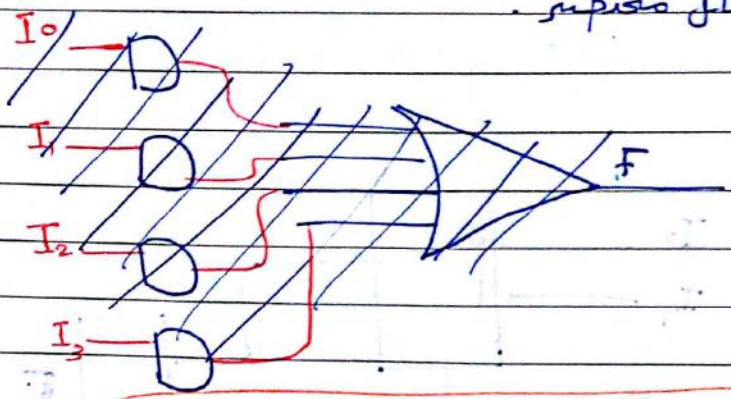
2017 / 2016

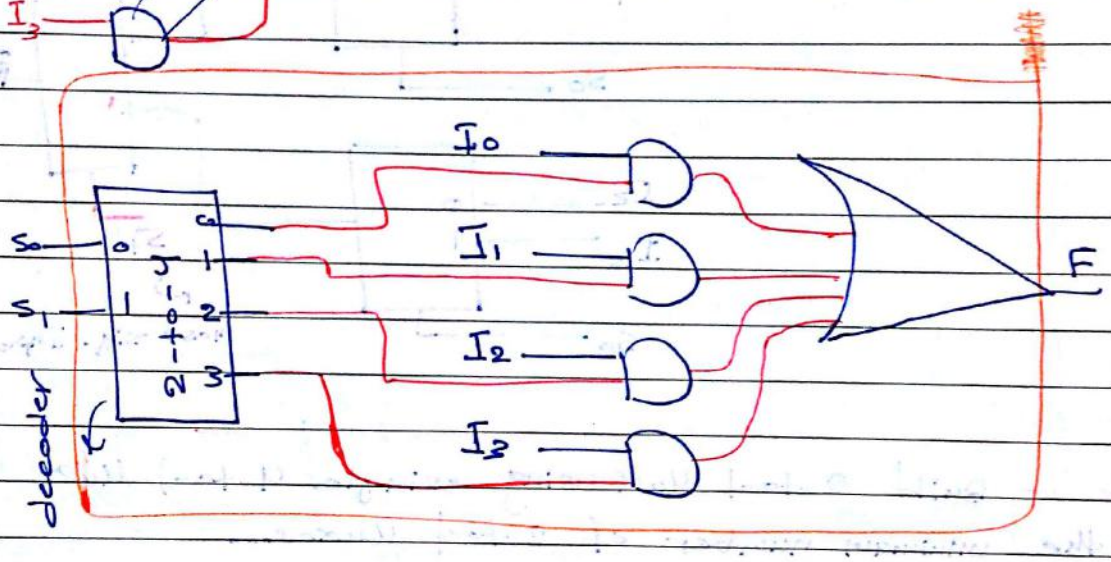
* 4-to-2 Mux



S_1	S_0	F
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

كيف يشتغل الشكل فوق.





Homework: 8-to-1 Mux → 3 and gate, 3 input OR gate, 3-to-8 decoder

16-to-1 Mux

↳ 16 2-Input AND

16-Input OR

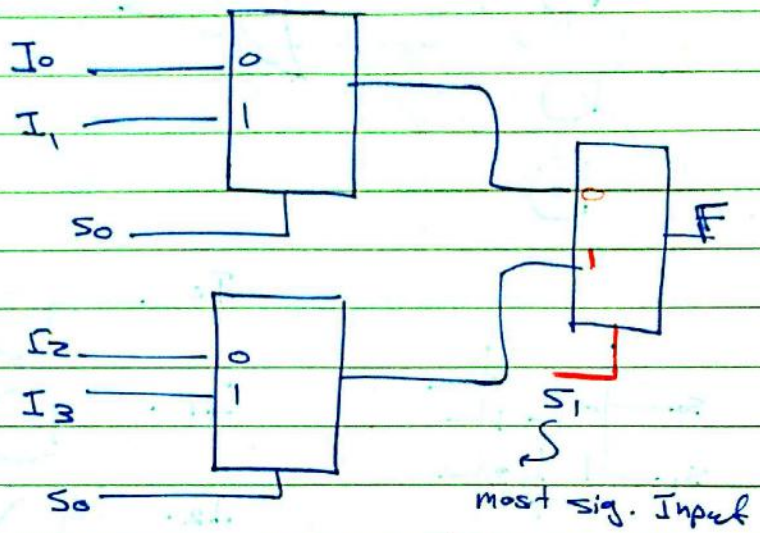
4-to-16 decoder

20 / 7 / 2016

* Building Large Muxes from smaller ones.

Ex:- Build 4-to-1 Mux using 2-to-1 Muxes only

S_1	S_0	
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

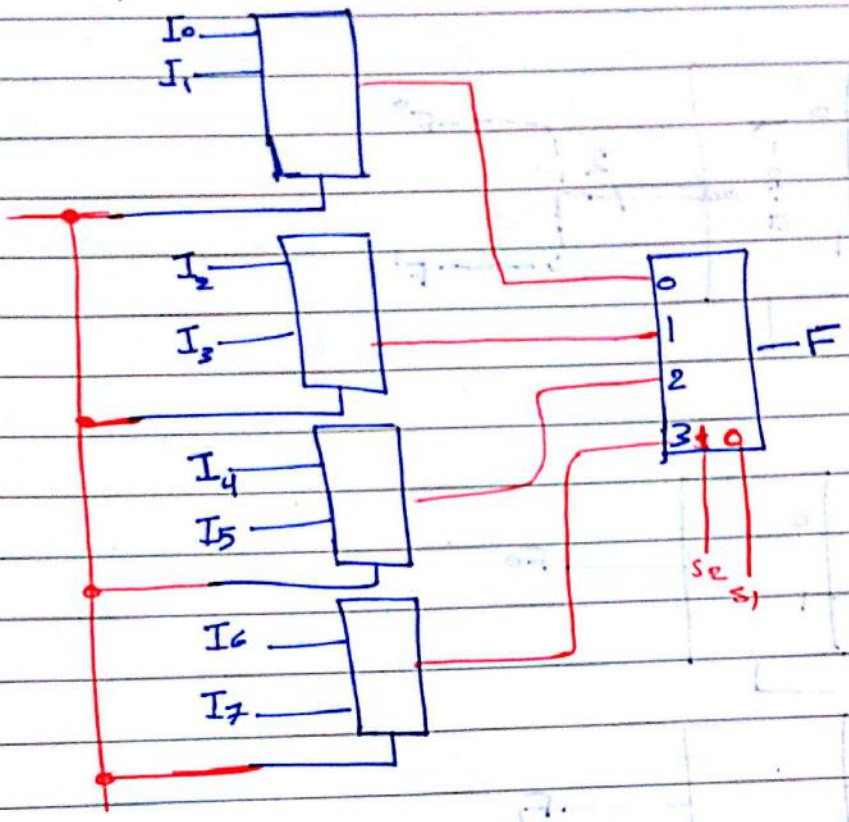


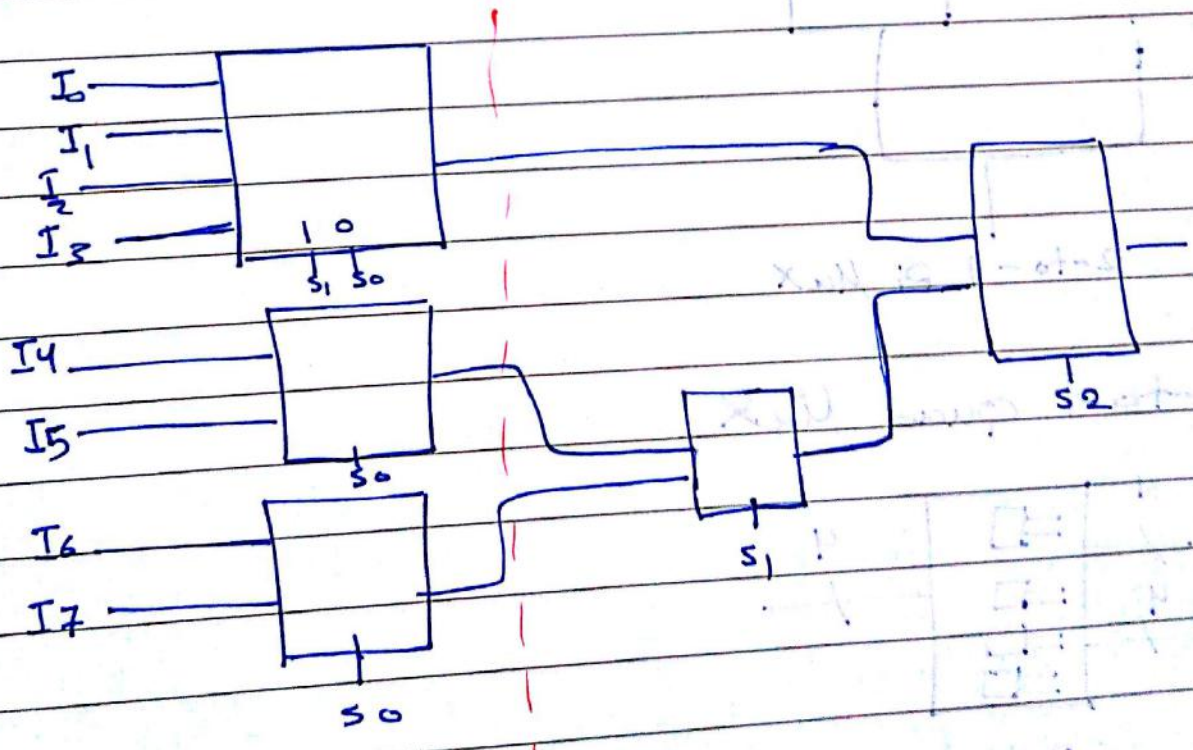
Example 8:- Build 2-to-1 Mux using a single 4-to-1 Mux and the minimum number of 2-to-1 Muxes...

MSB

S_2	S_1	S_0	F
0	0	0	I_0
0	0	1	I_1
0	1	0	I_2
0	1	1	I_3
1	0	0	I_4
1	0	1	I_5
1	1	0	I_6
1	1	1	I_7

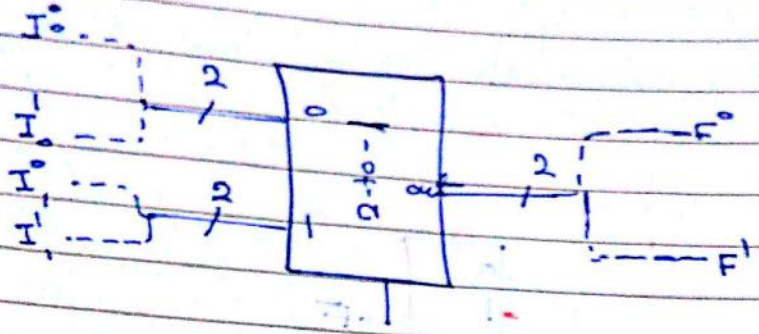
20/7/2016

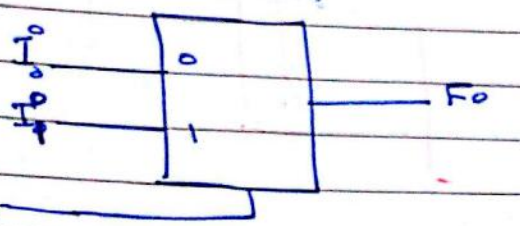


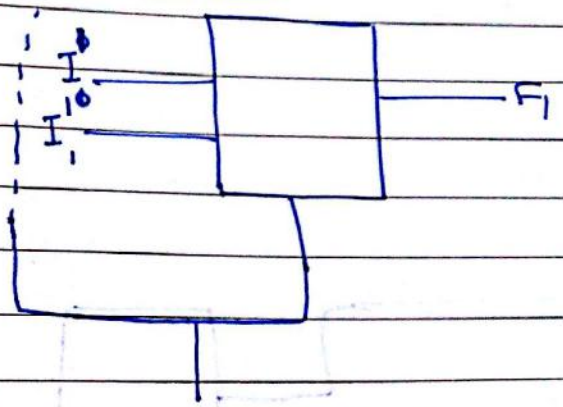


2017 /2018

* Multiplexer with expansion

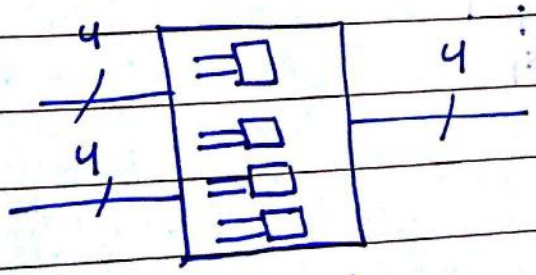






2-to-1 Bi MuX

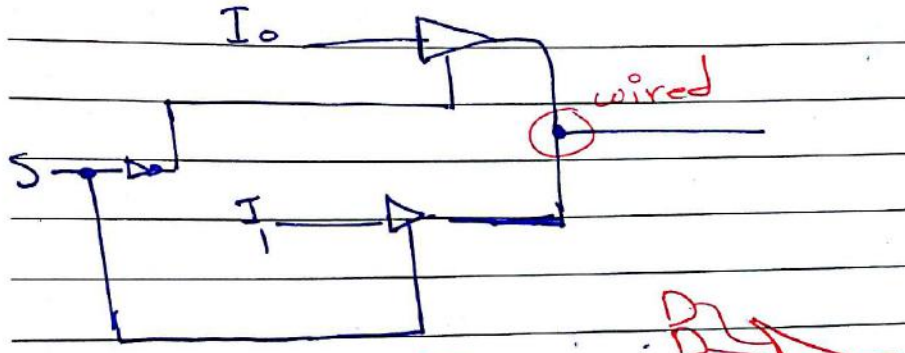
2-to-1 quad MuX

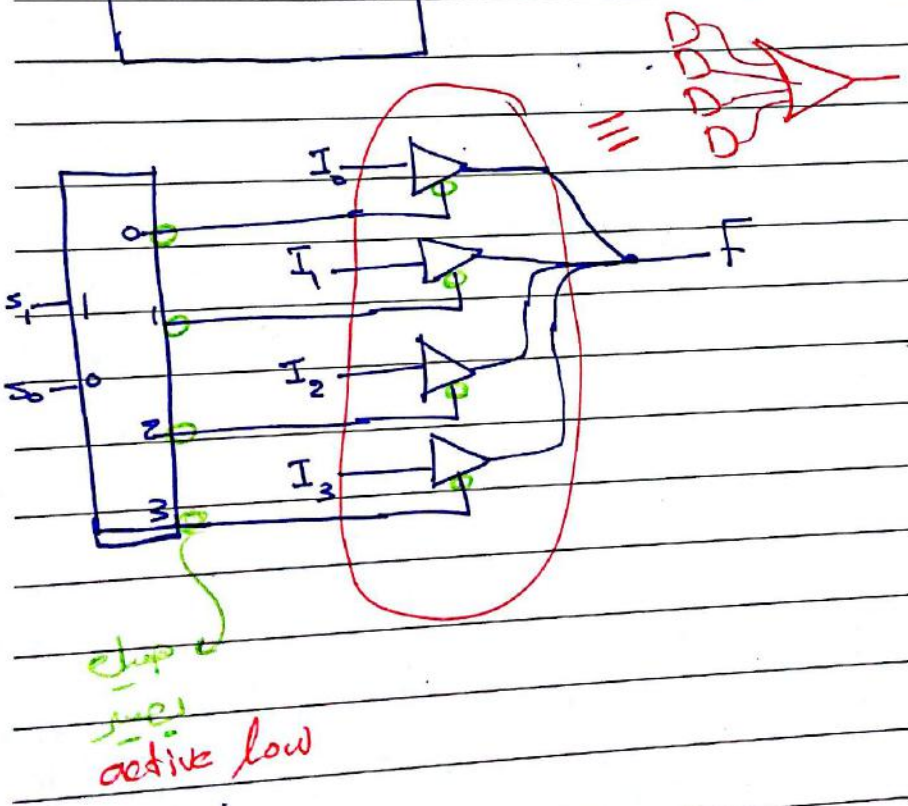


2-to-1
quad

* Implementing Muxes using Tri-state buffers & Decoders.

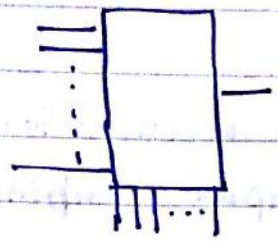
2-to-1





No. 24/7/2016

Implementing Logic functions using Muxes :-



* The Multiplexor can be consider as selection OR gate.

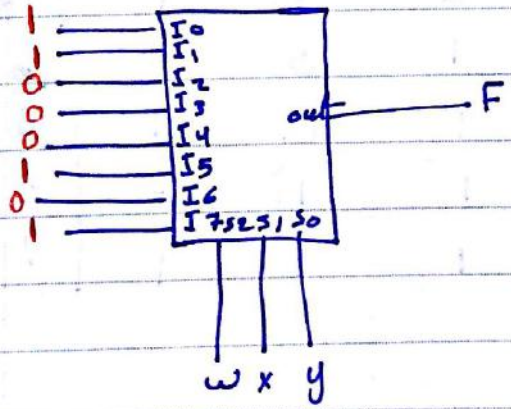
* we can Implement a function of N variable using N-to-1 Mux. only by :-

a) connecting the input variable to the select lines in order.

b) value-fix the inputs of the Mux based on the minterms or maxterms of the function.

Example :- $F(w, x, y) = \sum_m (0, 1, 5, 7)$

3-variables → 8-to-1 Mux



smile

No. 24/7/2016

⇒ However, when N increases, the size of the Mux increases dramatically...

⇒ We can reduce the cost by using an inverter...

(a) generate the T.T and arrange input combinations in order...

(b) use the N-1 input variables to partition the table into pairs of rows...

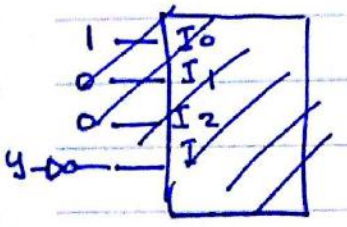
(c) for each pair, express the function in terms of the least significant variable...

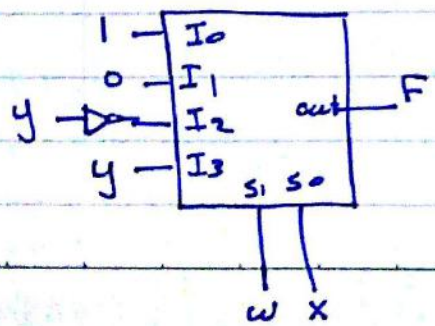
(D) connect the N-1 input variables to the select liner in order...

Example :- $F(w,x,y) = \sum_m (0,1,4,7)$

3-variables ⇒ 3-1 2-to-1 ⇒ 4-to-1 Mux

w	x	y	F	
0	0	0	1	F = 1
0	0	1	1	
0	1	0	0	
0	1	1	0	F = 0
1	0	0	1	F = ȳ
1	0	1	0	
1	1	0	0	F = y
1	1	1	1	



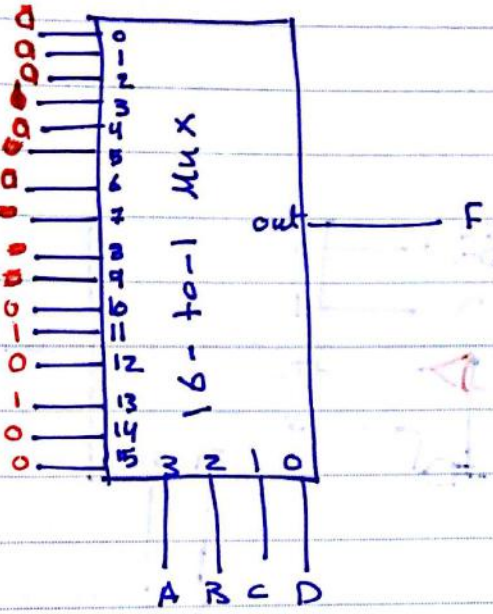


smile...

No 24/7/2016

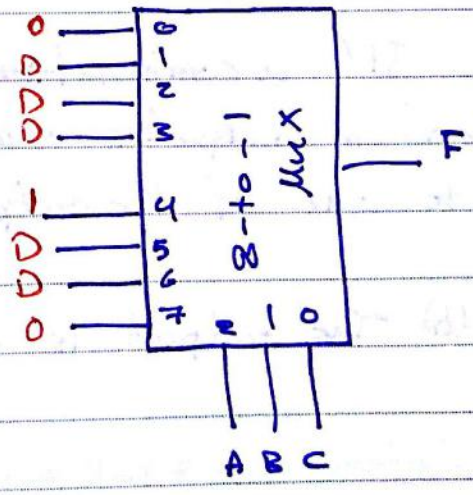
Example : $F(A,B,C,D) = \sum_M (0,1,2,4,6,10,12,14,15)$

4-variable → 2^4-to-1 mux



or using an inverter 4 variables → 8-to-1 mux.

	A	B	C	D	F	
0	0	0	0	0	0	F = 0
1	0	0	0	1	D	F = D
2	0	0	1	0	D	F = D
3	0	0	1	1	D	F = D
4	0	1	0	0	1	F = 1
5	0	1	0	1	D	F = D
6	0	1	1	0	D	F = D
7	0	1	1	1	0	F = 0
	1	0	0	0		
	1	0	0	1		
	1	0	1	0		
	1	0	1	1		
	1	1	0	0		
	1	1	0	1		
	1	1	1	0		
	1	1	1	1		

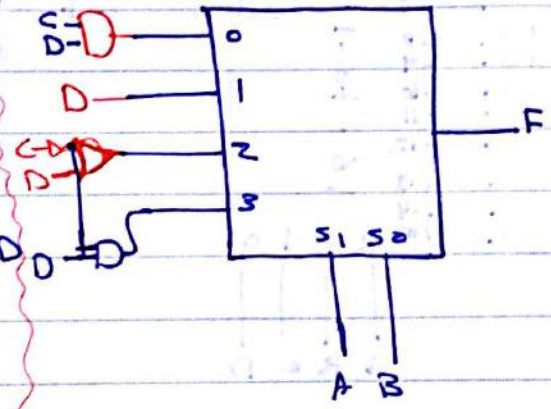


No. 24/7/2016

what if we have 4-to-1 Mux??

Group the rows based on the most significant 2-variable

A B	C D	F
0 0	0 0	0
	0 1	0 C.D
	1 0	0
	1 1	1
0 1		0
		1 D
		0
		1
1 0		1
		0 $\bar{C}$+D
		1
1 1		0
		1 $\bar{C}$.D
		0
		0

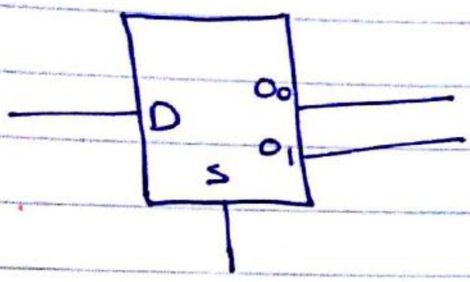


Example (D) DeMultiplexor (DMux)

It's a combinational logic circuit that has 1 Input and m outputs, such that (a) the input is routed/directed to one of the outputs.

(b) The output is specified using $\lceil \log_2 m \rceil$ select lines.

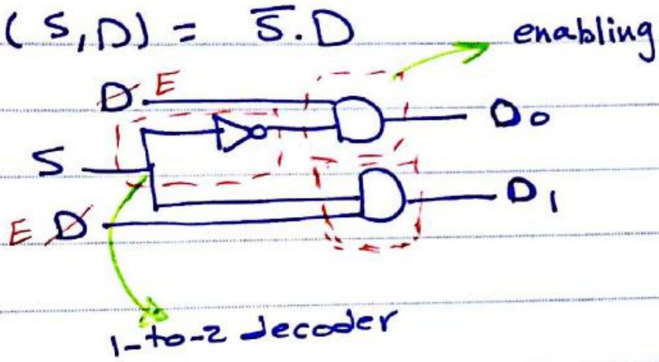
⇒ 1-to-2 DeMux



S	D	O_1	O_0
0	0	0	0
0	1	0	1
1	0	0	0
1	1	1	0

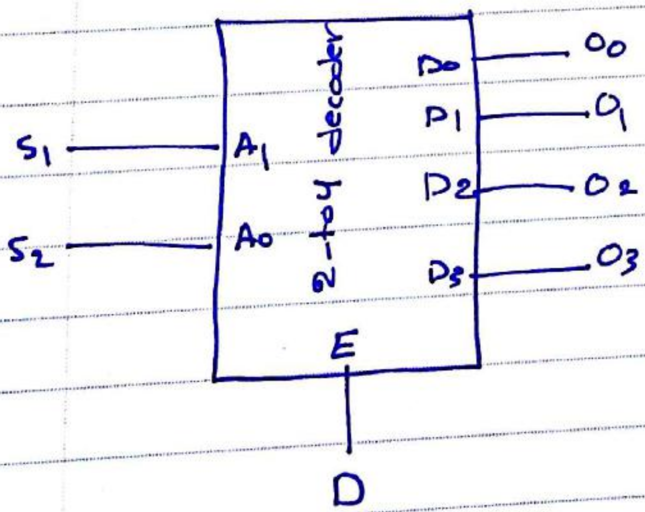
$O_1(S, D) = S \cdot D$

$O_0(S, D) = \bar{S} \cdot D$



1-to-2 deMux ≡ 1-to-2 decoder with enable...

⇒ 1-to-4 DeMux



إذا شبكنا ال Decoder
بهاي الطريقة ح يصير
DeMux

⇒ 1-to-n deMux ≡ log n-to-n decoder with enable...

No. 25/7/2016

Chapter 4 # Arithmatic Logic ...

4.1 Binary Arithmatic

* Addition

$$\begin{array}{r} 0 \\ +\ 0 \\ \hline 0 \end{array} \qquad \begin{array}{r} 0 \\ +\ 1 \\ \hline 1 \end{array} \qquad \begin{array}{r} 1 \\ +\ 0 \\ \hline 1 \end{array} \qquad \begin{array}{r} 1 \\ +\ 1 \\ \hline 10 \end{array}$$

(1 + 1 = 2)

carry = base

Decimal

$$\begin{array}{r} 3 \\ +\ 5 \\ \hline 8 \end{array} \Rightarrow \text{Binary} \quad \begin{array}{r} 1\,1\,1\,0 \leftarrow \text{carry} \\ +\ 011 \\ 101 \\ \hline 1000 \end{array}$$

8 ⟷ 1000 متساويات

* Subtraction

$$\begin{array}{r} 0 \\ -\ 0 \\ \hline 0 \end{array} \qquad \begin{array}{r} 1 \\ -\ 0 \\ \hline 1 \end{array} \qquad \begin{array}{r} 1 \\ -\ 1 \\ \hline 0 \end{array} \qquad \begin{array}{r} 0 \\ -\ 1 \\ \hline -1 \end{array}$$

⇓

$$\begin{array}{r} (1)\emptyset \\ -\ 1 \\ \hline 11 \end{array}$$

وقفنا هاد لواحد لما استعرنا ← (1)∅

مثل كأنها 2

No. 25/6/2018

-3 Decimal ⇒ Binary 11

2 -10

1 01

-5 Binary

6 ⇒ 101 ⇒ 110

-1 -110 -101

1111 ?? ← ما طلع الجواب صحيح !! $\ominus 001$

عكسناهم وطرحنا وطلع الجواب بالسالب

⇒ How to deal with th sign ?!

4.2 Signed numbers

⇒ we need to ~~defined~~ define anotation or approach to represent signed numbers ...

⇒ we can add an extra bit for the sign

⇒ Tuo general approaches :-

1) Signed magnitude ...

2) signed complement ...

(A) Signed-Magnitude

sign (1 bit) | magnitude (n-1)

⇒ sign = { 0, +Ve ; 1, -Ve

No. 25/7/2016

→ sign bit

0 ← (+)3

1 ← (-)3

Example : list all 3-bit signed numbers that are represent using (SM) signed magnitude

S	M			
A_2	A_1	A_0	Value	
0	0	0	+0 ??	
0	0	1	+1	Valid +
0	1	0	+2	
0	1	1	+3	
1	0	0	-0 ??	
1	0	1	-1	Valid -
1	1	0	-2	
1	1	1	-3	

* problems :-

① Two representations for '0' ...

② Hardwar Implementation more complex ...

Decimal	SM
+3	011
+ -3	111
0	1000 +2 ??

① : given an n-bit signed number in SM notation, what is the Maximum value? what is the minimum value?

smile for life

No. 25/7/2016

* max $= +(2^{n-1}-1)$
* min $= -(2^{n-1}-1)$

(B) Signed complement...
- Diminished Radix complement ($\tilde{N}$)
- Radix complement (N^*)

* Diminished Radix complement

For an n-digit number N in base 'r', the diminished Radix complement $\tilde{N}$ is given by :-

$$\tilde{N} = r^n - 1 - N$$

Example:- what is $\tilde{N}$?

a. $(18)_{10} \longrightarrow r=10 \quad n=2$

$\tilde{N} = 10^2 - 1 - 18 = 81 \Rightarrow N + \tilde{N} = 99$

b. $(23)_{10} \Rightarrow \tilde{N} = 10^2 - 1 - 23 = 76 \Rightarrow N + \tilde{N} = 99$

9's complement

c. $(01)_2 \longrightarrow r=2 \quad n=2$

$\tilde{N} = 2^2 - 1 - 1 = 2 \Rightarrow N + \tilde{N} = (11)_2$

d. $(101)_2 \longrightarrow r=2 \quad n=3$

$\tilde{N} = 2^3 - 1 - 5 = 2 \longrightarrow N + \tilde{N} = (111)_2$

1's complement

smile for life

No. 25/7/2016

If we assume that $r^n - 1$ is our zero, How to know whether $\tilde{N}$ represents a negative or positive number?

solution

⇒ Add a sign digit before computing the complement

* $(18)_{10} \rightarrow (018)_{10}$ (sign)

$\tilde{N} = 10^3 - 1 - 18 = (981)_{10}$ (-ve) $\rightarrow N + \tilde{N} = (999)_{10}$

* $(101)_2 \rightarrow (0\,101)_2$ (sign)

$\tilde{N} = 2^4 - 1 - 5 = (10)_{10} = (1010)_2 \Rightarrow N + \tilde{N} = (1111)_2$ (*-1)

signed comp. value $(1010)_2 \rightarrow 2^4 - 1 - 10 \Rightarrow 5 = (0101)_2$ signed 1's complement.

Example: List all 3-bit signed numbers when signed 1's complement is used.

sign A_2	A_1	A_0	value	
0	0	0	0	+ve
0	0	1	1	
0	1	0	2	
0	1	1	3	
1	0	0	-3	-ve
1	0	1	-2	
1	1	0	-1	
1	1	1	-0	

$\Rightarrow \tilde{N} = 2^3 - 1 - 6 = 1 = (001)_2$

$\Rightarrow \tilde{N} = 2^3 - 1 - 7 = 0 = (000)_2$

smile

No. 25/7/2016

Problem 5 :-

① Two Zeros ?!

② HW.

+3 / +2 / +1 → is → 011 / 110 / *001 ??

* −3 / +2 / −1 → is → 100 / 010 / 0110 ✓

* في بعض الحالات الجواب مباشرة بطلع صح بس أحياناً بيي إضافة الـ carry وارجع أعدله مع الجواب بطلع غلط !!

* +3 / + −2 / +1 → 011 / 101 / (1)000 → 1 / 001

what is the max value of a n-bit signed 1s complement number ?! what is the minimum ??

$max = 2^{n-1} - 1$

$min = -(2^{n-1} - 1)$

Shortcut :- the 1's complement of a number is obtained by flipping individual bits.

$(101110)_2 = ??$ → unsigned $= (46)_{10}$

→ SM $= (-14)_{10}$

→ 1s $= -(17)_{10}$

No. 25/7/2016

$(0110111)_2$ ⇒ unsigned $= (55)_{10}$

⇒ SM $= (55)_{10}$ → لأن الرقم موجب

⇒ 1's comp. $= (55)_0$

إذا كان الرقم موجب لا تواجه مشكلة ...

* Radix complement :-

The radix complement for an n-digit number N in base r is obtained by

$$N^* = r^n - N$$

Example :- compute N^*

a. $(18)_{10}$ → r=10 n=2

$N^* = 10^2 - 18 = (82)_{10}$ → 18 + 82 = 100

} 10's complement

b. $(101)_2$ → r= 2 n=3

$N^* = 2^3 - 5 = 3 = (011)_2$ → $N + N^* = (1000)_2$

لأنه carry تجاهل ال 3 bit هنا

} 2's complement

* How to know whether (011) is +3 or -5 !!

⇒ add sigh bit !!

smile

No. 25/7/2016

$(0101)_2$ ← sign → $r=2$, $n=4$

$N^* = 16 - 5 = 11 = (1011)_2$

$N^* + N = (0101)_2 + (1011)_2 = (*0000)_2$ ← نهمل

Example:- list all 3-bit numbers using signed 2's complement approach

	A_2	A_1	A_0	value	
+ve	0	0	0	+0	
	0	0	1	+1	
	0	1	0	+2	
	0	1	1	+3	
-ve	1	0	0	-4	المميزة ⇒ $2^3 - 4 = 4$
	1	0	1	-3	⇒ $2^3 - 5 = -3$
	1	1	0	-2	⇒ $2^3 - 6 = 2$
	1	1	1	-1	⇒ $2^3 - 7 = 1 = (001)_2$

* Note:- there is only 1 representation for the zero !!

$\begin{array}{r} +3 \\ -2 \\ \hline 1 \end{array}$ ⟹ 2's → $\begin{array}{r} 011 \\ \phi 10 \\ \hline *001 \end{array}$ ✓ الجواب صح إذا أهملنا الـ carry !!

⇒ Simpler Hw. Implementation ...

smile

No. 25/7/2016

* For n-bit signed 2's comp. number

$max = 2^{n-1} - 1$

$min = -2^{n-1}$

* shortcuts to compute 2's comp.

① $N^x = \tilde{N} + 1$

N^x → 2's, $\tilde{N}$ → 1's

Example: $+3 \xrightarrow{2's}$ using 3 bits ↙ 011 $\xrightarrow{1's}$ 100 $\xrightarrow{+1}$ $(101)_2 \rightarrow$ (-3)

② Scaned the number from right to left, keep all zero's and the first 1, then flip remaining bits

17

$+3 \xrightarrow{2's} ??$

011

101

No. 26/7/2016

* $(1100100)_2$ → unsigned $= (100)_{10}$

→ signed → SM = ~~$(-38)_{10}$~~ $(-36)_{10}$

→ 1's comp = $(-27)_{10}$

→ 2's comp. = $(-28)_{10}$

0011100 (بضل انزل الاصفار لحد أول واحد بعدها ببلش أعمل comp.)

باخذ ال 1's وبجمعلها ①

0011011
\+ 1
\-------
0011100

نفس الجواب

Ex:- Show the representation of the following numbers in SM, 1's and 2's complement assume 5 bits always.

Number	SM	1's	2's
+12	01100	01100	01100
−7	10111 (1 = sign, 0111 = mag)	00111 → الموجب 11000 → comp	00111 11001
−11	11011	01011 → الموجب 10100 → comp.	01011 → الموجب 10101 → comp.
19 (Max = $2^{n-1} - 1$, $2^4 - 1 = 15$ → أعلى قيمة)	X (ما بقدر أمثلها)	X	X
−16	X	X	10000 ← موجب 16 10000 → comp. Min = 2^{n-1}

Note :- For any n-bit signed 2's complement number the Decimal value is given By

sign

$b_{n-1}\, b_{n-2} \ldots b_0 \,.\, b_{-1} \ldots b_{-n}$

$$\text{value} = \sum_{k=-n}^{n-2} 2^{k} \cdot b_k + (-1)^{b_{n-1}} * 2^{n-1} * b_{n-1}$$

الرقم نفسه اللي هو الـ bit

-11

$(11010 1)_2 \rightarrow 5 + (-1)^1 * 2^4 * 1$

$5 - 16 = -11$

$(01100)_2 \rightarrow 12 + 0 = 12$

4.3 2's Complement Arithmetic

(A) Addition

add corresponding bits from the two numbers and ignore the carry out of the MSB

Example:- $15 + -20 \rightarrow -5$ => 2's compl بدنا نفضل بالحالة

* لو ماحدد عدد الـ bits بلزم أكون عارفة عدد Min لا bits اللي أنا بحتاجهم

$-20 \rightarrow 010100 \xrightarrow{x-1} 101100$ (+20 → −20)

$15 \rightarrow 01111$

أعمل شي بدي 6 bits

```
   11
0 ← 001111 +
    101100
  --------
 ✗  111011 ——→ 2's → (000101)₂ → (+5)
```

كل هاي الخطوة

لحتى أتأكد من الحل.

No. 26/7/2016

ⓑ Subtraction :-

$10 - 13 = 10 + (-13)$

* To compute $M - N$ using 2's complement, compute $M + (-N)$

EX:- $10 - 13 = -3$

$(10)_{10} \rightarrow (01010)_2$

$(-13)_{10} = (01101)_2 \xrightarrow{2's} (10011)_2$ ← $(-13)_{10}$

$$\begin{array}{r} 01010 \\ +\,10011 \\ \hline 11101 = -3 \end{array}$$

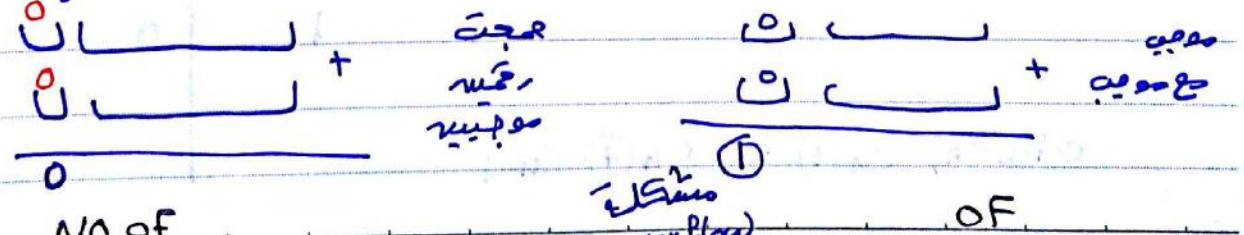
ⓒ over flow :-

⇒ overflow occures when the result cant be represented using the available number of bits.

⇒ How to detect overflow?!

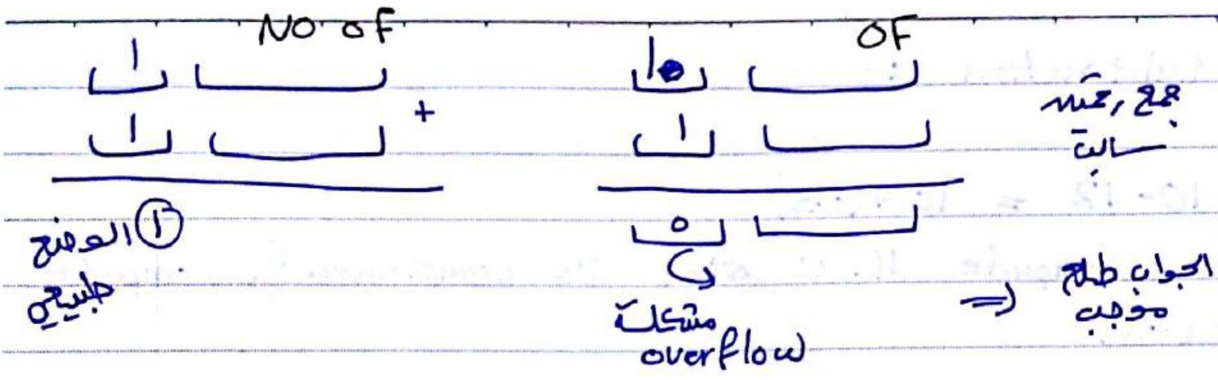
(a) for unsigned numbers, overflow is detected when the carry out of the MSB is = 1

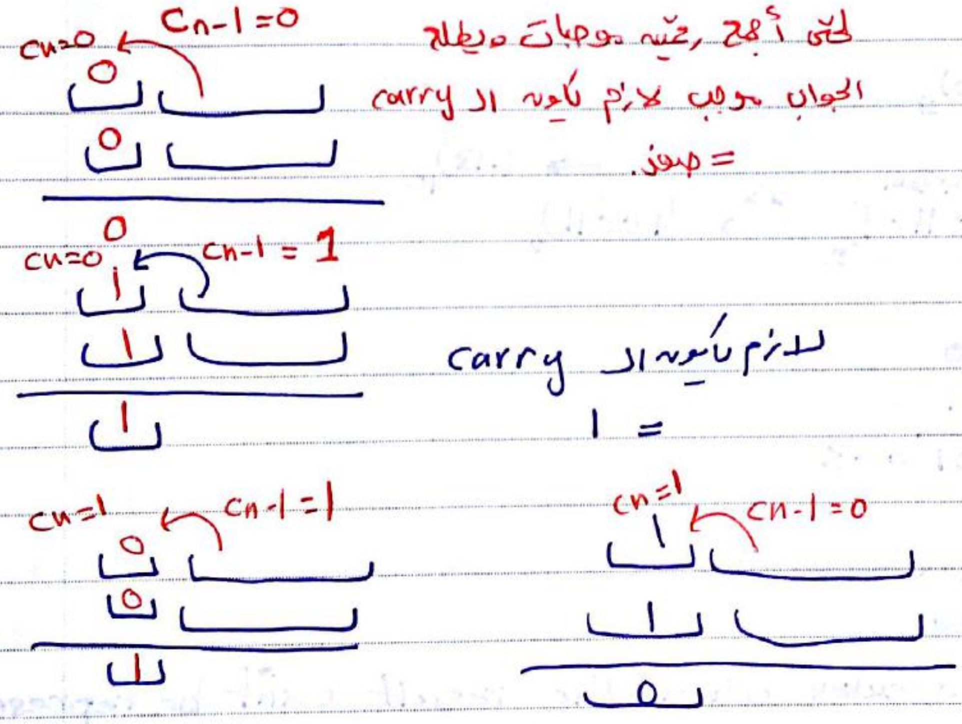
(b) signed numbers



smile for life

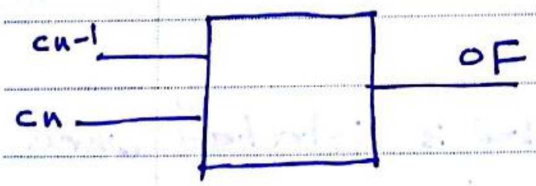
No. 26/7/2016





لما أجمع رقمين موجبات وطلع الجواب موجب لازم تكون ال carry = صفر

لازم تكون ال carry = 1



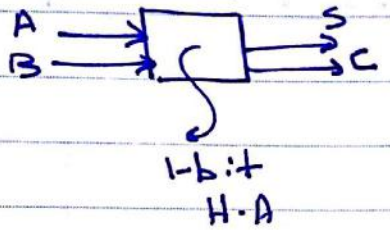
Cn	Cn-1	OF
0	0	0
0	1	1
1	0	1
1	1	0

$OF(C_n, C_{n-1}) = C_n \oplus C_{n-1}$

No. 26/7/2016

4.4 Arithmatic Hardware ...

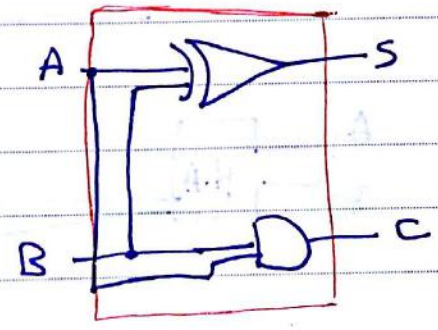
(A) 1-bit half adder (HA)



A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

$S(A,B) = A \oplus B$

$C(A,B) = AB$



(B) 2-bit adder

⇒ Traditional approach → HW → $A_1 A_0$ $B_1 B_0$ $S_1 S_0$ C

x??

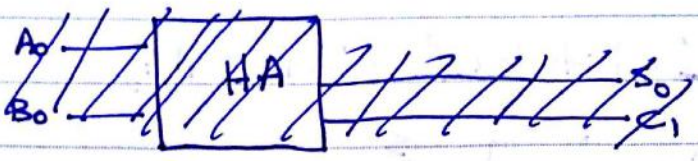
⇒ consider 64-bit adder ?!

```
  1 1 1
  0 1 0 1
  0 0 1 1  +
-----------
  0 1 0 0 0
```

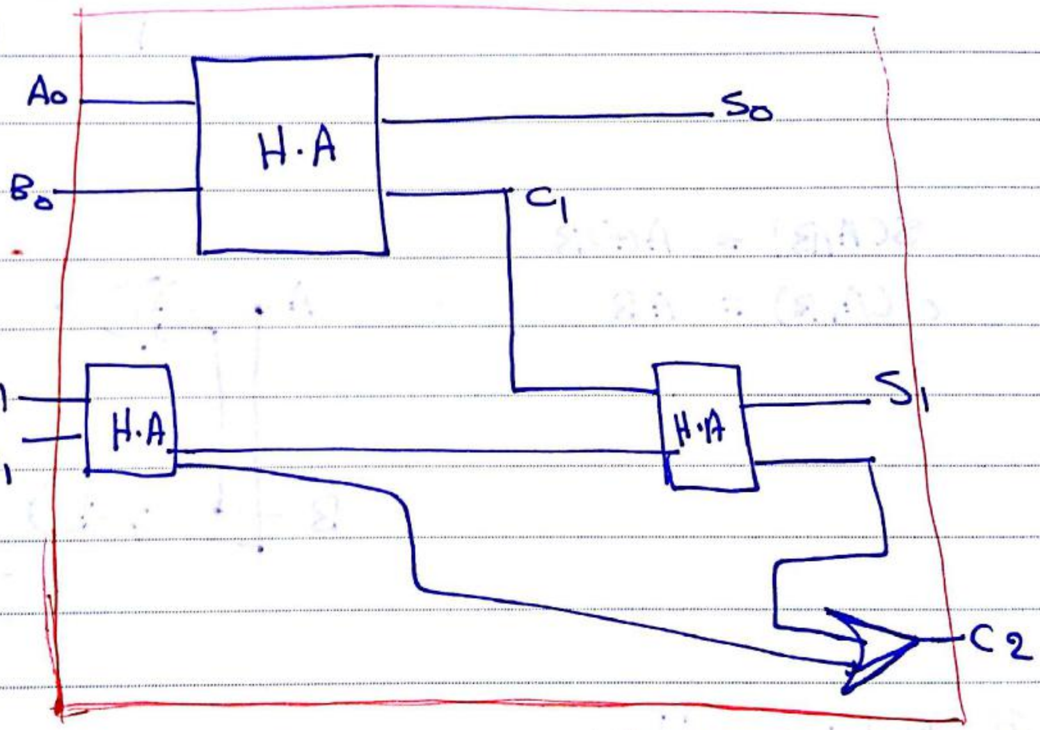
carry

No. 26/7/2016

$$\begin{array}{r} C_2 \; C_1 \\ A_1 A_0 \\ + \; B_1 B_0 \\ \hline C_2 S_1 S_0 \end{array}$$



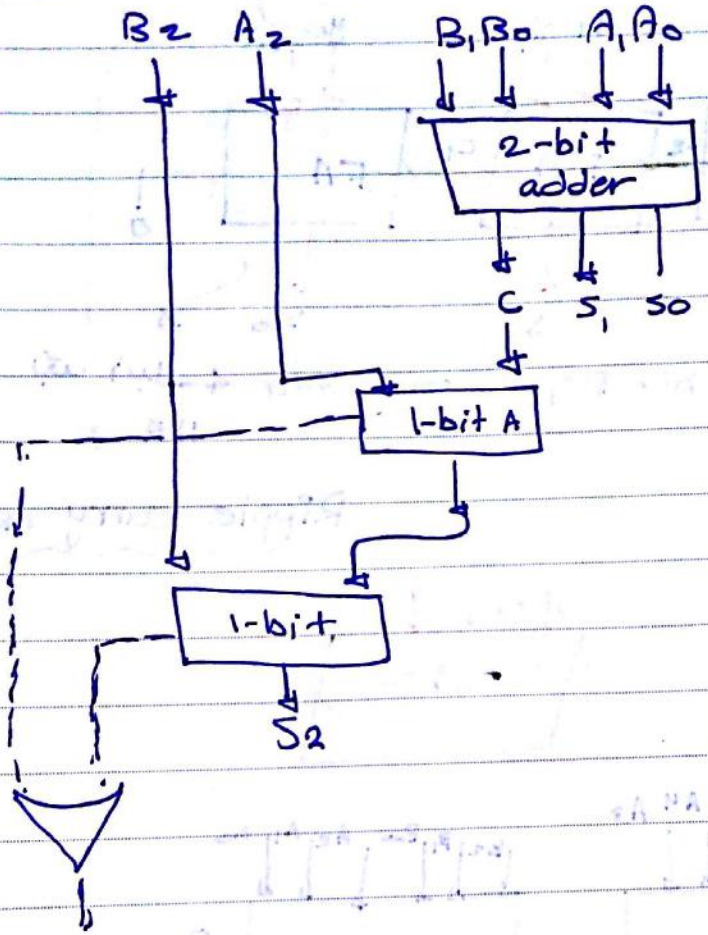
2-bit adder



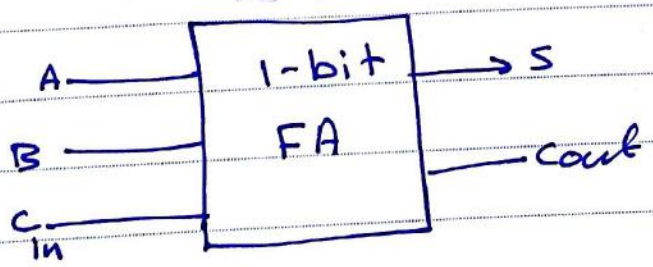
HW: 4-bit adder !!

* ع تقنع أسئلة من OR gate ...

No. 26/7/2016



© 1-bit full adder (FA) ③ දෙපැදිය



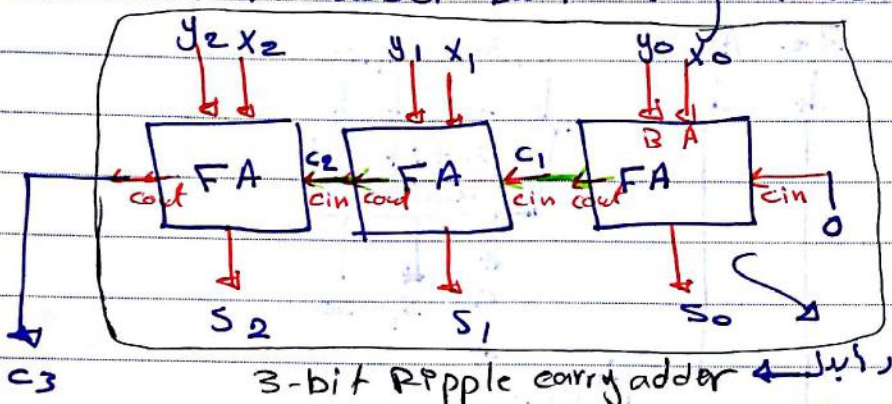
A	B	Cin	S	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$S = A \oplus B \oplus C$ ⇒ K-Map කරලා

$Cout = ACin + Bcin + AB$

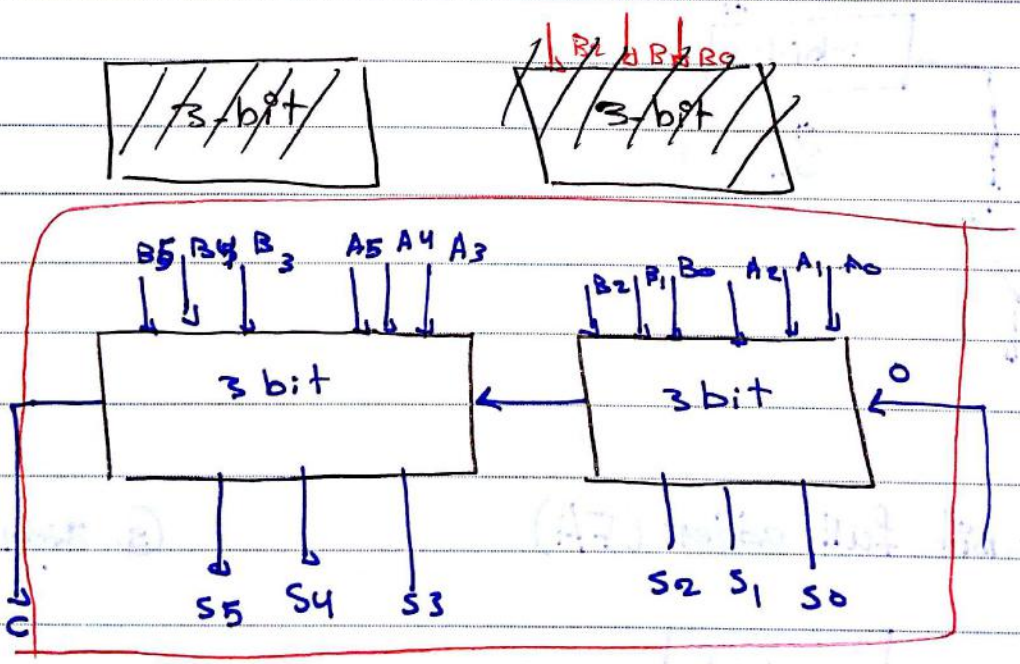
No. 26/7/2016

How about 3-bit adder built using FAs only ?!



3-bit Ripple carry adder ← بكدر أبدل HA ←

Ripple carry adder



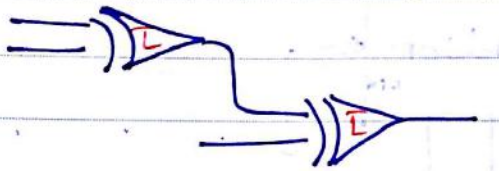
No. 27/7/2016

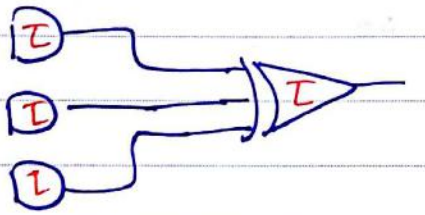
RCA :-

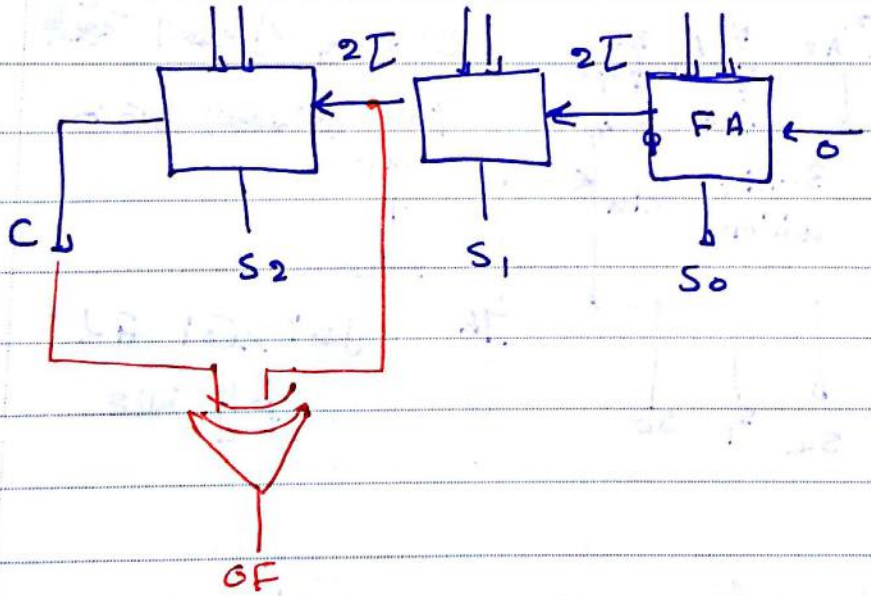
$S = A \oplus B \oplus C_{in}$

$C_{out} = AB + B C_{in} + A C_{in}$

T = delay ⇒ সময়







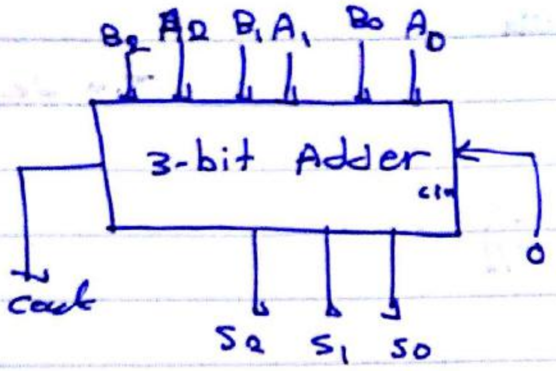
6T

64 * 2T = 128T

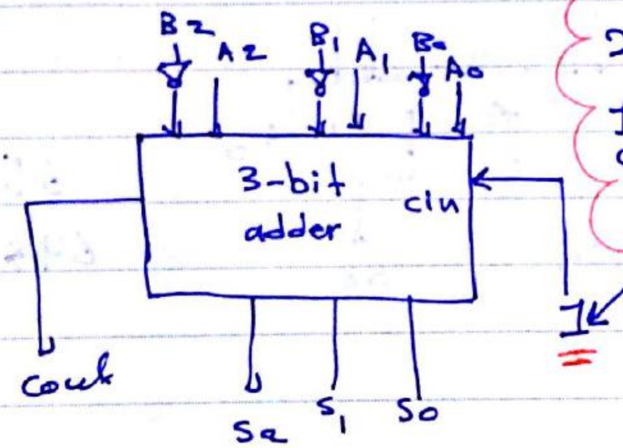
No. 27/7/2016

D Subtractor!

$M - N = M + (-N)$ $A - B = A + (-B)$



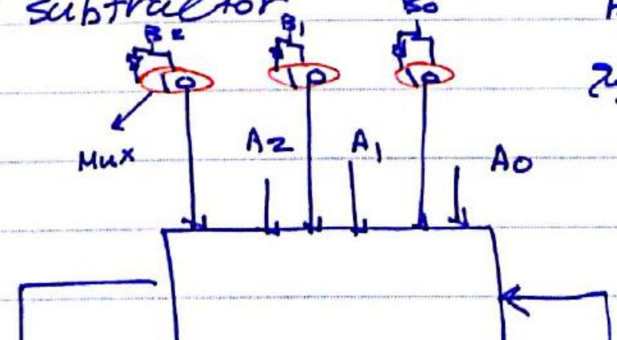
⇓ 2's comp.



طريقة التحويل إلى 2's Comp
من قبل الحصول على
1's Comp
+
1

لحتى أقدر أعمل
عملية الطرح

E Adder / subtractor

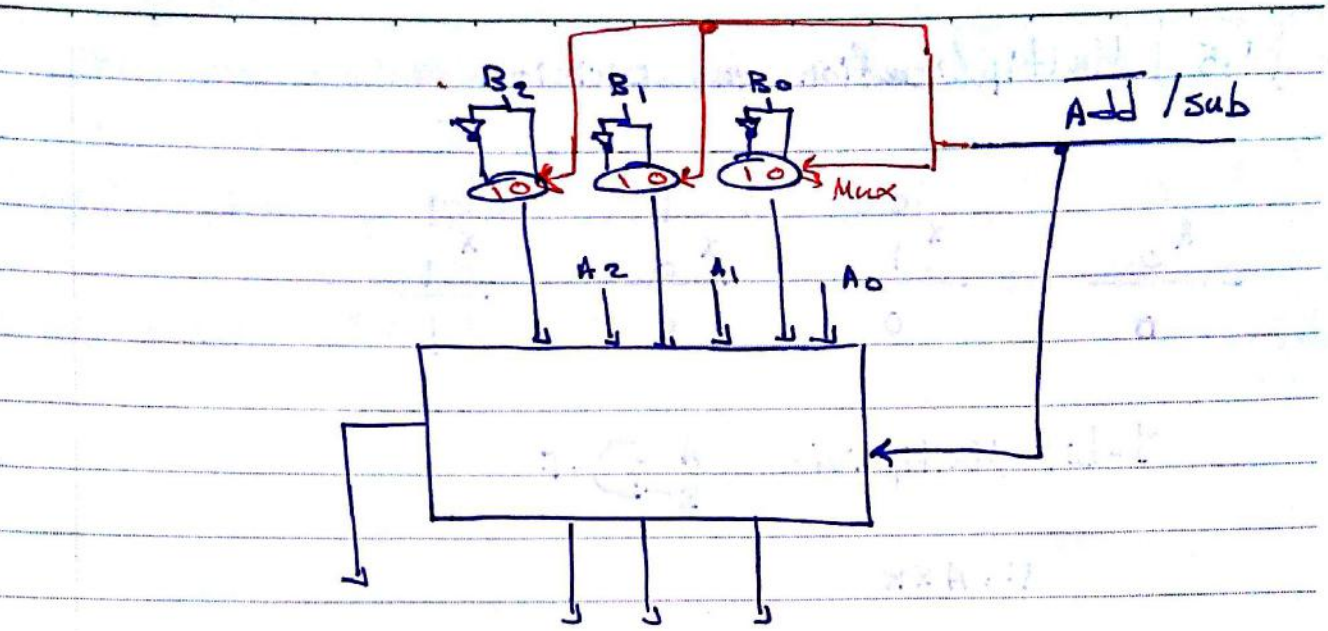


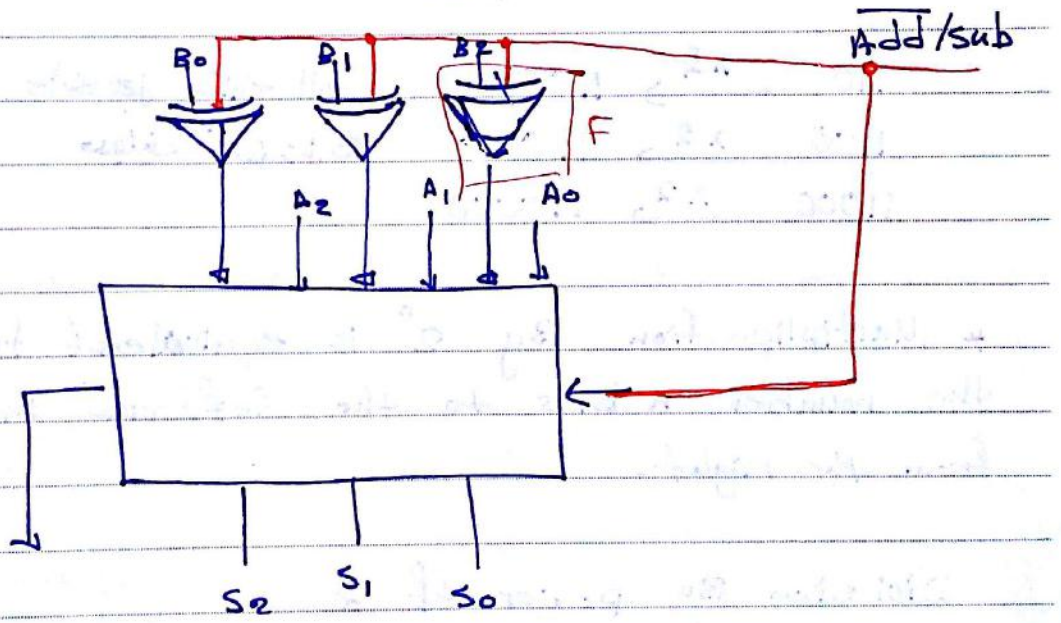
Add / sub.

إذا كانت القيمة (0) بعمل عملية جمع
وإذا كانت 1 بعمل عملية
طرح.

← الرسمة
ناقصة
الرسمة الكامل
→
بتيع

No. 27/7/2016





Bx	Add/sub	E
0	0	0
0	1	1
1	0	1
1	1	0

smile for life

No. 27/7/2016

4.5 Multiplication and Division ...

$$\begin{array}{r} 0 \\ \times\ 0 \\ \hline 0 \end{array} \qquad \begin{array}{r} 0 \\ \times\ 1 \\ \hline 0 \end{array} \qquad \begin{array}{r} 1 \\ \times\ 0 \\ \hline 0 \end{array} \qquad \begin{array}{r} 1 \\ \times\ 1 \\ \hline 1 \end{array}$$

1-bit Multiplier 

$F = A \times B$

(A) Multiplication by power of 2

(6) 110 $\xrightarrow{\times 2}$ 1100 (12)

(12) 1100 $\xrightarrow{\times 2}$ 11000 (24)

11000 $\xrightarrow{\times 2}$ 110000

كأنني بجعل الرقم shift left وبضيف صفر زيادة

* Multiplication By 2^n is equivalent to shifting the number n bits to the left and inserting zeros from the right.

(B) Division By power of 2

11000 $\xrightarrow{\div 2}$ 01100

1100 $\xrightarrow{\div 2}$ 00110

110 $\xrightarrow{\div 2}$ 00011

11 $\xrightarrow{\div 2}$ 00001

5bit ليه ↵

No. 27/7/2016

for unsigned numbers, division by 2^n is equivalent to shifting the number by n bits to the right and inserting zeros from the left.

هنا غلط عشان 5 bits

11000 (−8) $\xrightarrow{\div 2}$ 01100 ??

غلط

الصفر هون يعني سالب

⇐ لما بدي أعمل قسمة لازم أحافظ على إشارة الرقم.

11100 (−4) $\xrightarrow{\div 2}$ 11110 (−2)
(sign)

11110 (−2) $\xrightarrow{\div 2}$ 11111 (−1)
(sign)

ملخص

⇒ for signed numbers, division by 2^n is equivalent to shifting the number to the right by n bits and replicating the sign.

0 11001 ÷4 → 000110

100100 ÷8 → (3) bits ← بيعملهم shift مكانهم ومن ثم بنحط ... 3 bits

111100

−28 ÷ 8 = −4

No 27/7/2016

© Multiplication By constant :-

B X 101

→ 4 bits (HW)

B : { B_3 B_2 B_1 B_0 }

→ 7 bits

B_3 B_2 B_1 B_0 XXX	F_6 ---- F_0
	0 ——— --
	— ——— — 101
	— — — — 1010
	— — — —

B_3 B_2 B_1 B_0

1 0 1 X

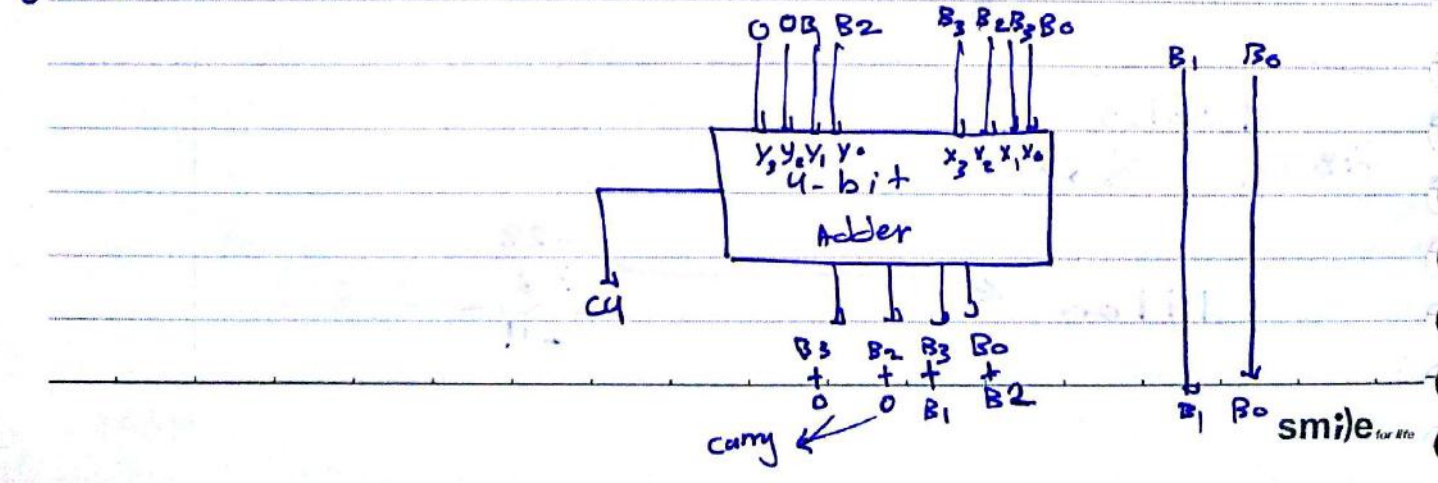
B_3 B_2 B_1 B_0

0 0 0 0 0 0 +

B_3 B_2 B_1 B_0 0 0

B_3 B_2 B_3 B_0 B_1 B_0

B_3 + 0 B_2 + 0 B_3 + B_1 B_0 + B_2



smile for life

No. 27/7/2016

$B_3\ B_2\ B_1\ B_0$ → (HW)

1 1 0 X

4.6 Data with extension

* Hardware units have fixed size for their inputs.

* However, the operands size might be smaller than the size of the input?!

* This implies that the size of the operands should be extended to ~~match~~ match the size of the input.

How!!

* Two cases :-

① zero extension

0100

0000 | 0100

This works perfectly with usigned numbers.

② sign extension

1010 ≡ -6

0000 | 1010 +10 zero ext. X

1111 | 1010 -6

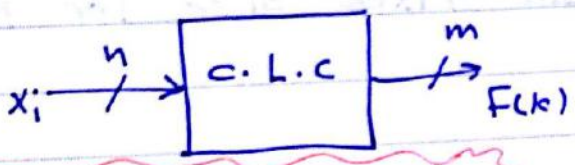
No. 31/7/2016

ch.5 Sequential circuits

5.1 Introduction

Digital logic circuits → combinational logic circuit
→ sequential logic circuit

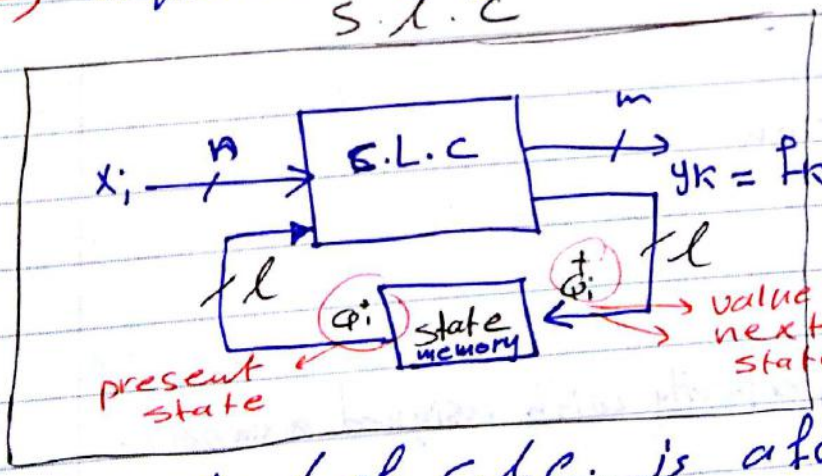
* combinational logic circuits



$y_k = f_x(x_0, \dots, x_{n-1})$

At any instant of time, the output is function of the curent input only

⇒ Sequential logic circuits



$Q_i^+ \rightarrow Q_i(t+1)$ } مجرد طريقة للتعبير
$Q_i \rightarrow Q_i(t)$ }

The output of S.L.C. is a function of the curent input in addition to value stored in the state element.

smile

No. 31/7/2016

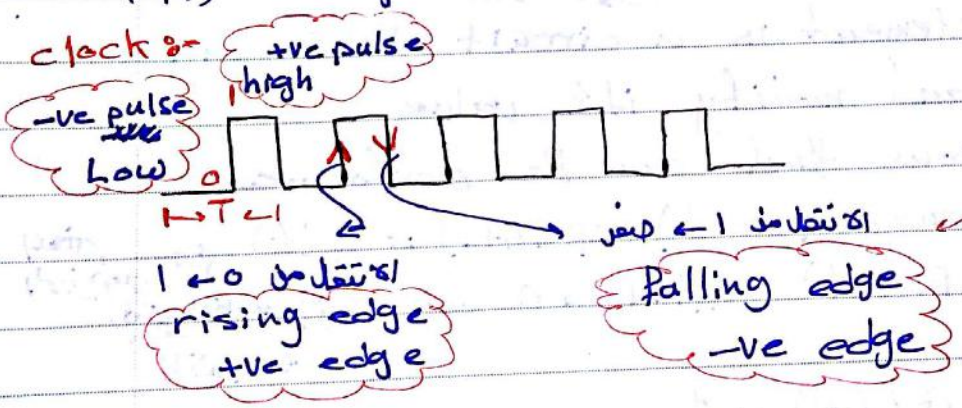
$$y_k = f_k(x_0, \dots, x_{n-1}, Q_0, \dots Q_{l-1})$$

$$Q_i^+ = f(x_0, \dots x_{n-1}, Q_0, \dots Q_{l-1})$$

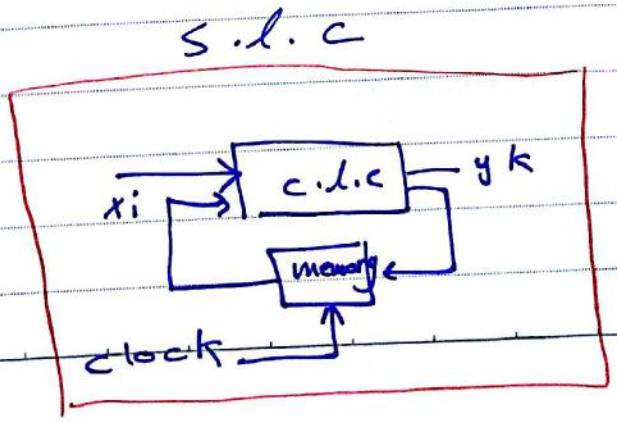
present state

* sequential circuits :-
  - ↳ Synchronous ← متزامن
  - ↳ Asynchronous

* synchronous sequential circuits :-
→ the operation of the circuit in terms of changing the outputs (y_k) and the present state (Q_i) is synchronized with a clock. (time)
(Special events)



→ So, y_k and Q_i are allowed to change during last some time



smile

No. 31/7/2016

⇒ simpler to analyze and design...
⇒ slower !!

* Asynchronous sequential circuits
  ⇒ No clock !!
  ⇒ the stored value (Q_i) is allowed to change at any time...

  ⇒ Faster !!
⇒ But more complex to analyze and design...

5.2 Storge Elements...

* A storge element is a logic circuit that can store 1 bit and modify it's value...
* The operations that can be performed
  ⇒ No change (Hold) $Q(t+1) = Q(t)$ ⇐ يبقى الرباطين
  ⇒ clear (Reset) $Q(t+1) = 0$ ⇒ يقوم بتقريب القيمة القديمة
  ⇒ Set $Q(t+1) = 1$
  ⇒ complement $Q(t+1) = \overline{Q(t)}$

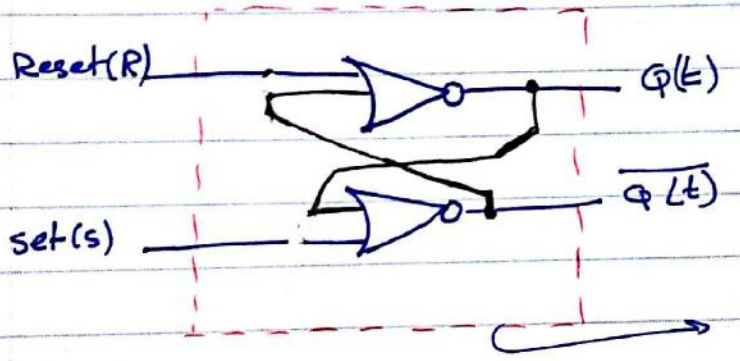
No. 31/7/2016

(A) Latches :-

⇒ Storge element that can store one bit indefinitly as long as power is applied...

→ It has inputs to specify and modify the stored value...

1.) cross-coupled NOR gate latch...



time

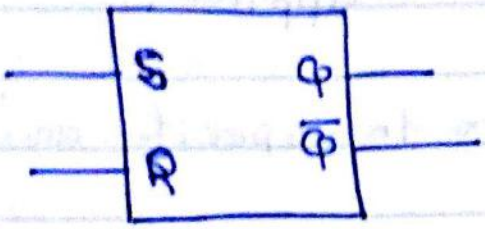
	S	R	Q(t)	$\overline{Q(t)}$	
t_0	x	x	x	x	
t_1	1	0	1	0	set
t_2	0	0	1	0	No change (Hold)
t_3	0	1	0	1	→ reset
t_4	0	0	0	1	No change (Hold)
t_5	1	1	0	0	⇒ ?? undefined state.

not allowed ✗

الدخل التالي الحالي
0, 0 → No change
1, 0 → set
0, 1 → reset

No 31/7/2016

SR Latch ⇒ graphical ال simpale



مكافئة الرسمة يلي بالخلف

S	R	Q(t)	$Q^+(t)$ → Q(t+1)	
0	0	0	0	no change
0	0	1	0	
0	1	0	0	Reset
0	1	1	0	Reset
1	0	0	1	set
1	0	1	1	
1	1	0	X	undefined
1	1	1	X	

$Q(t+1)$

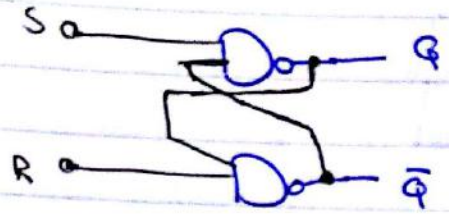
S \ R	0	1	0	0
S	1	1	X	X

Q

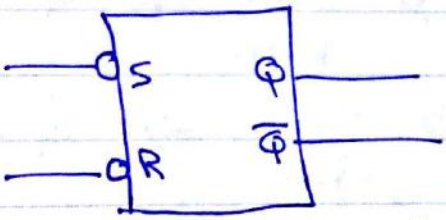
$$Q(t+1) = S\bar{R} + \bar{R}Q(t)$$

No. 1/8/2016

2. cross couple NAND gates ($\bar{S}\bar{R}$ Latch)



	S	R	Q	$\bar{Q}$	
t_0	x	x	x	x	
t_1	0	1	1	0	⇒ set
t_2	1	1	1	0	⇒ Hold
t_3	1	0	0	1	⇒ Reset
t_4	1	1	0	1	⇒ Hold
t_5	0	0	1	1	⇒ undefined



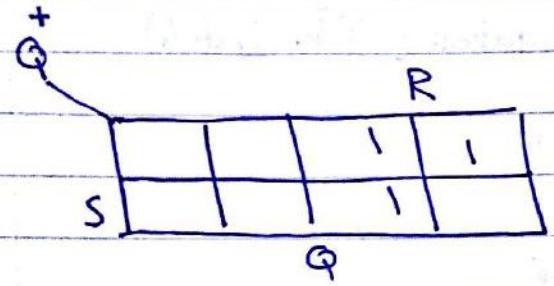
* SR latch with active low input
* $\bar{S}\bar{R}$ latch

$Q(t+1) = Q^+$??

S	R	Q	Q^+
0	0	0	undefined
0	0	1	undefined
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

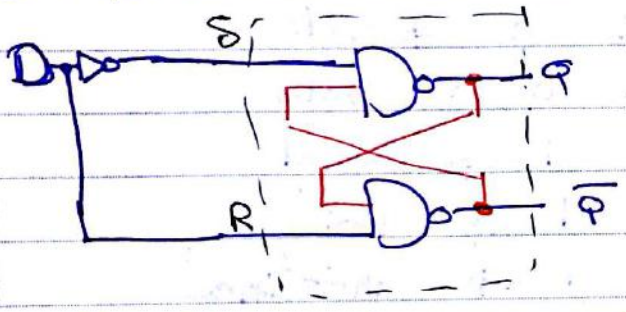
بسم الله → smile

No. 1/8/2016



$Q^+ = QR + \bar{S}R$

3. D - Latch

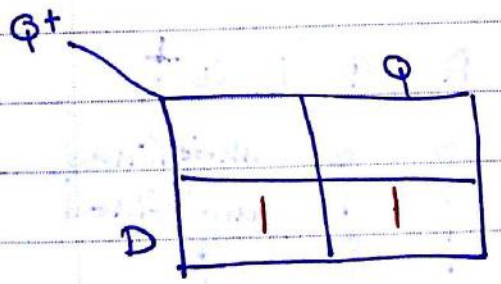


$\begin{cases} S = \bar{D} \\ R = D \end{cases}$ في حالة الطريقة بخزن في الـ Q إما 1 أو (zero) 0

$D=0 \Rightarrow S=1, R=0 \Rightarrow Q=0$

$D=1 \Rightarrow S=0, R=1 \Rightarrow Q=1$

$Q^+ = Q(t+1) = D$



D	Q	Q^+
0	0	0
0	1	0
1	0	1
1	1	1

smile

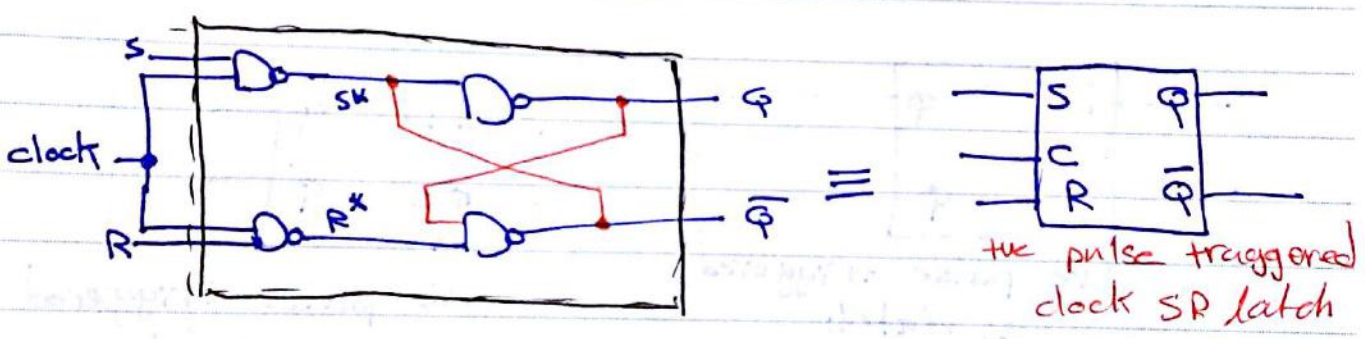
No. 1/8/2016

B. clocked Latches :

* All the latches we discussed previously are Asynchronous is the case that they are allowed to change the stored value as soon as the input changes.

clock ai:
* However it's desirable sometime to change the state (stored value) at some time.

1. clocked SR Latch

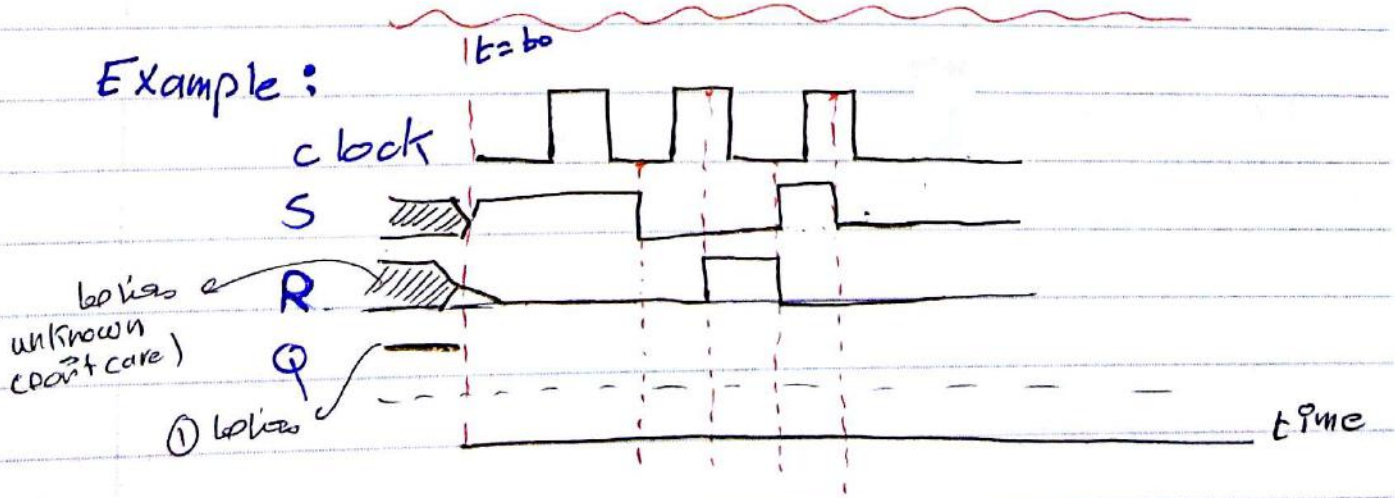


clock = 0 $\Rightarrow S^* = R^* = 1 \Rightarrow Q^+ = Q \rightarrow$ no change

clock = 1 $\Rightarrow S^* = \bar{S}$ $\Rightarrow$ normal SR Latch

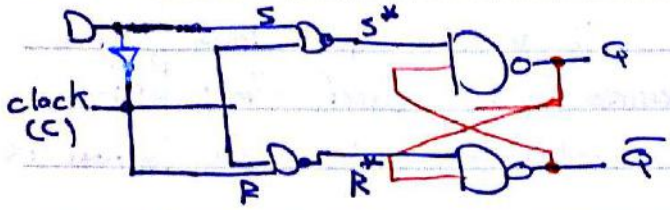
$R^* = \bar{R}$

Example:



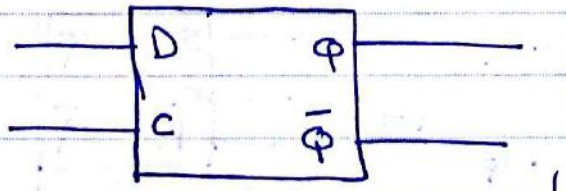
No 1/8/2016

2. clocked D-latch

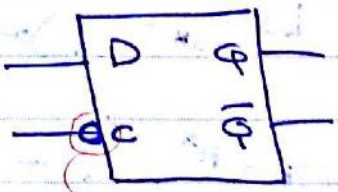


- $C = 0 \rightarrow S^* = R^* = 1 \Rightarrow Q^+ = Q$

  $C = 1 \Rightarrow S^* = \bar{S} = D \Rightarrow$ Normal operation
  $R^* = \bar{R} = \bar{D}$ ↳ $Q^+ = D$



+ve pulse triggered D-Latch



-ve pulse triggered D-Latch

Example :-

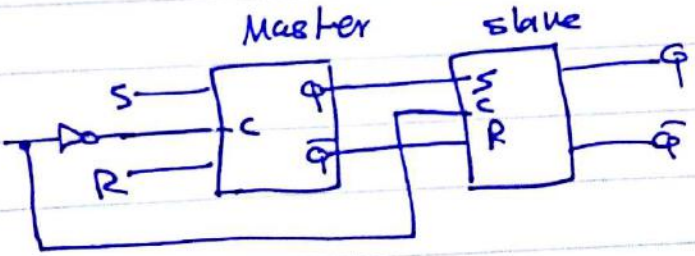
No 1/8/2016

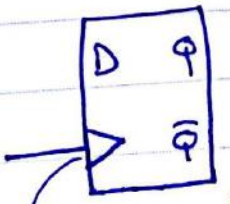
C. Flip-Flops :-

* Storage elements that are allowed to change the stored value at spacific time only.

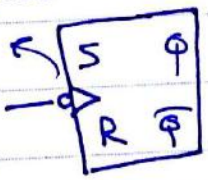
spacific time → Rising edge (+ve)
spacific time → falling edge (-ve)

* Implementation :-
① Master-slave
② Edge-triggered





→ +ve edge triggered D.FF



-ve. edge triggered SR FF

No. 1/8/2016

Example : given a(-ve) edge triggerred SR FF complete the diagram.

No 2/8/2016

Storage elements :-

↓

Latches

(SR, $\bar{S}\bar{R}$, D)

Asynch. (allowed to change the stored value at any instant of time).

⇓

clocked Latches

(SR, D)

synch (with clock input)

The change in the stored value is allowed during +ve pulse and -ve pulse of the clock.

⇓

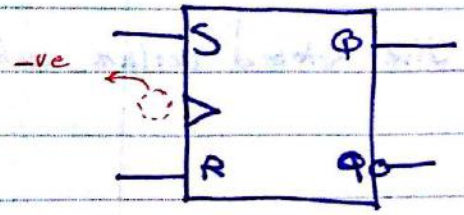
flip flops (SR, D) => synch.

(The change of the stored value is allowed at ↑ or ↓ of clock).

No. 2/8/2016

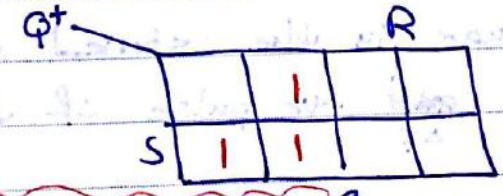
5.3 More on FFs :-

1. SR FF



S	R	Q(t+1)
0	0	Q(t)
0	1	0
1	0	1
1	1	x x x

* characteristic table



$$Q(t+1) = Q^+ = S\bar{R} + Q(t)\bar{R}$$

The excitation table for the SR FF is :-

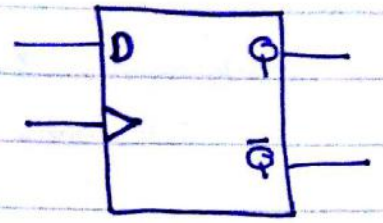
Q(t)	Q(t+1)	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

0 → 0 : Hold
0 → 1 : set
1 → 0 : reset

smile for life

No. 2/3/2016

2. D FF

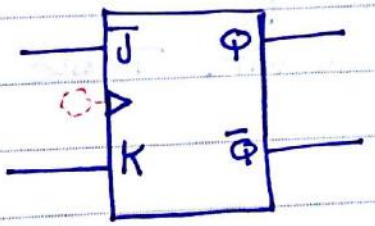


D	Q(t+1)
0	0
1	1

char. table

Q(t)	Q(t+1)	D
0	0	0
0	1	1
1	0	0
1	1	1

3. JK FF

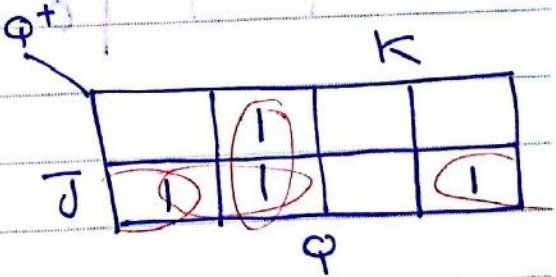


J	K	Q^+
0	0	Q
0	1	0
1	0	1
1	1	$\bar{Q}$

⇒ char. Table

complement / Toggle

J	K	Q	Q(t+1)
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0



$Q(t+1) = J\bar{Q} + \bar{K}Q$

smile for life

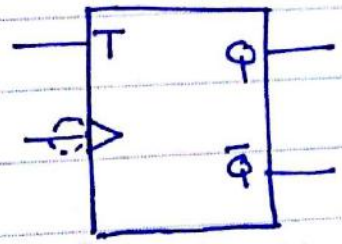
No. 2/12/2016

* Excitation :-

Q	Q^+	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

4. T FF

Toggle



T	Q^+
0	Q ⇒ Hold
1	$\bar{Q}$ ⇒ Toggle

Char. Table

T	Q	Q^+
0	0	0
0	1	1
1	0	1
1	1	0

$Q^+ = T \oplus R$

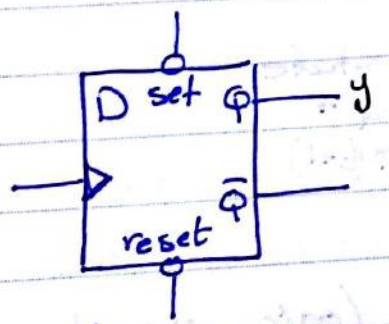
No. 2/8/2016

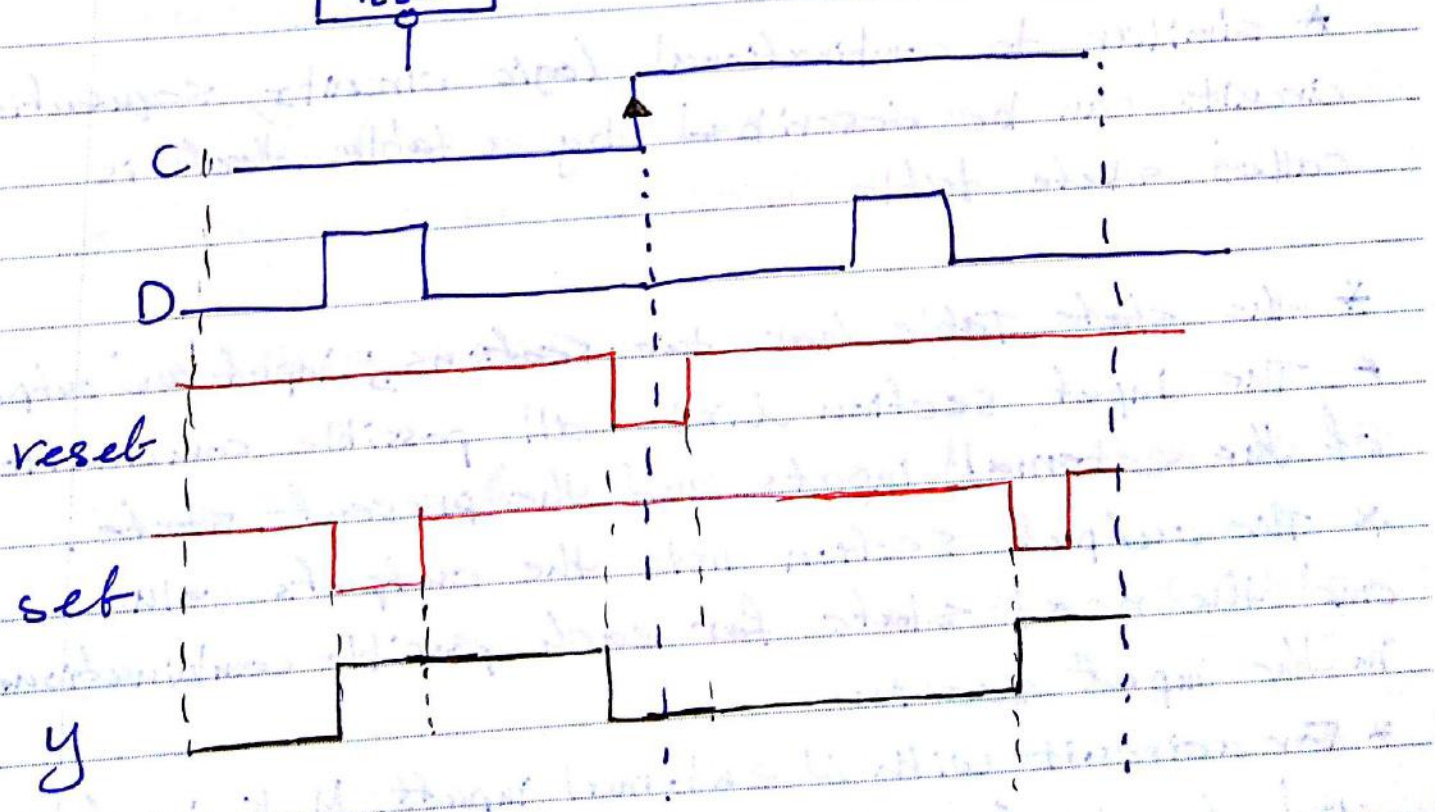
5.4 FFs with direct inputs :-

* Some time we need to change the stored value in the FF without waiting for the edge.

⇒ In other words, we want to change the stored value as soon as possible.

* The direct Inputs are additional FF inputs that can reset or set the FF irrespective of the clock.



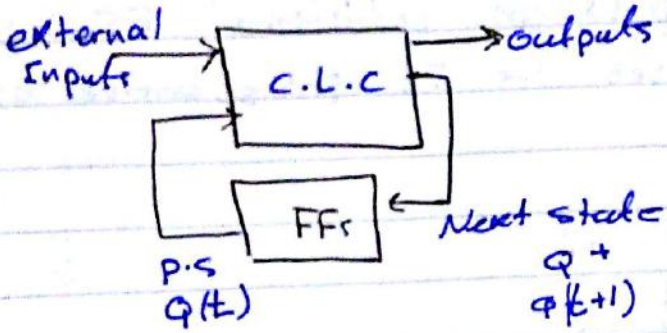


smile for life

No 2/3/2016

5.5 Analysis of synch. Seq. Circuits.

* Given a sequential circuit we want to determine the behavior of the circuit overtime with respect to any change in the input and the present state (stored value) ...



* similar to combinational logic circuits, sequential circuits can be described by a table that is called state table ...

* the state Table has two sections; input and output
* The input section lists all possible combinations of the external inputs and the present state.
* The output section lists the outputs value and the next-state for each possible combination in the input section
* For a circuit with N external inputs, M K-input FF and L outputs, the size of the state table

smile

No. 2 / 8 / 2016

is :-

عدد الأعمدة في out.

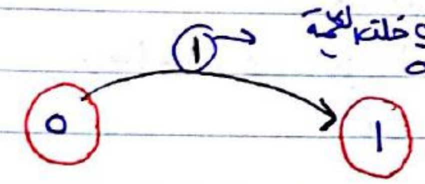
$$2^{N+M} * (l + M + N+M + M \times K)$$

عدد الأعمدة في ال Input — مدخلات FF

* The sequential circuit can also be described By a state diagram. which is a graphical representation of the state table such that:-

(a) Each possible state is represented by a circle...

(b) The transition Between states is represented By an arrow



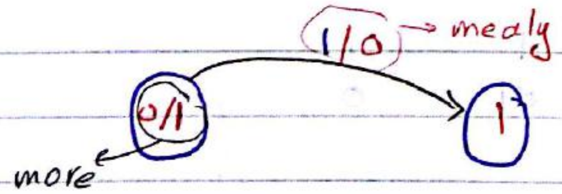
That is labled By the input value that caused the transition.

(c) The outputs are placed on the arrow if the circuit is Mealy machine or inside the circuits if it's a Moore machine...

اسم واحد

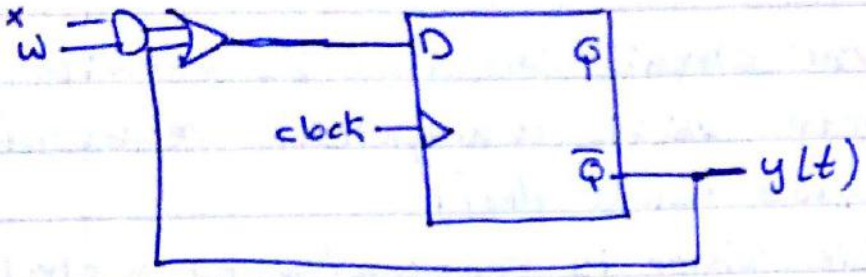
Mealy → outputs are function of Input and ps

Moore → " " " " ps only.



2/23/2016

Example1 - Derive the state table and diagram for the following circuit...



$N = 2 \rightarrow$ w and x

(1) FF $\Rightarrow M = 1 \Rightarrow$ 2 states

$l = 1 \rightarrow y(t)$

$K = 1$

$\Rightarrow 2^{N+M} * (M+N + M+l + K*M)$

$= 8 * 6$

FF Input equation

$D = w.x + \overline{Q(t)}$

$y(t) = \overline{Q(t)} \Rightarrow$ Moore

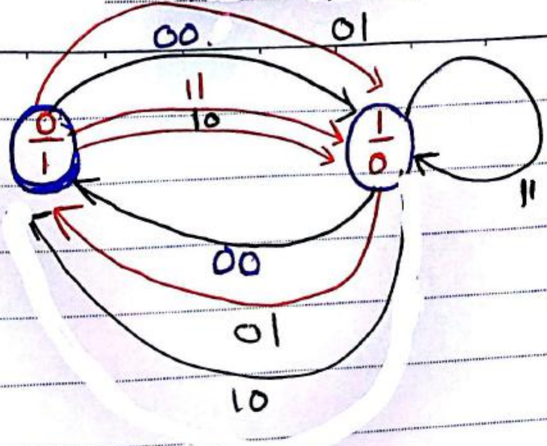
↳ FF output equation.

external input w	external input x	PS Q	D	NS Q(t+1)	output y(t)
0	0	0	1	1	1
0	0	1	0	0	0
0	1	0	1	1	1
0	1	1	0	0	0
1	0	0	1	1	1
1	0	1	0	0	0
1	1	0	1	1	1
1	1	1	1	1	0

smile

2/8/2016

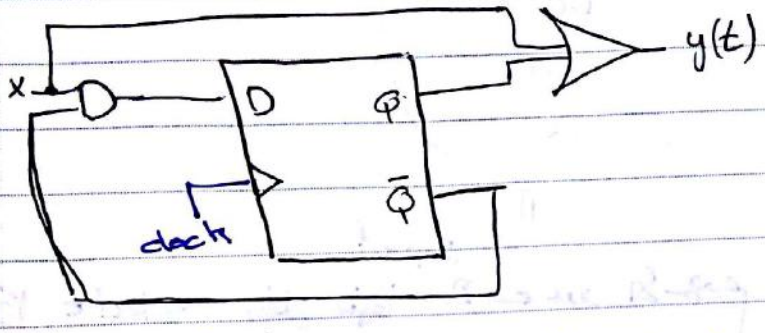
No.



كل state بطلع منها 2 Input ـ عدد الأحتمالات

No. 3/2/2016

Example:- Determine the state table and diagram for the following circuit...



Inputs = 1 $\Rightarrow$ N = 1 $\Rightarrow$ x

FF = 1 $\Rightarrow$ M = 1 $\Rightarrow$ Q (state) (two states)

output = 1 $\Rightarrow$ L = 1 $\Rightarrow$ y

FF type is D $\Rightarrow$ K = 1

$2^{(N+M)} * (N+M+L+M+M*K)$

$2^2 * (1+1+1+1+1)$

$= 4 * 5$

PS Q	Input X	D	NS Q^+	output y
0	0	0	0	0
0	1	1	1	1
1	0	0	0	1
1	1	0	0	1

$Q^+ = D$

FF Input equation

$D = x \cdot \bar{Q}(t)$

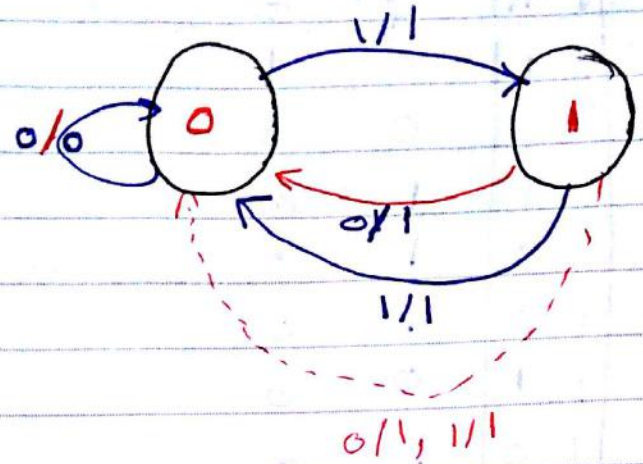
output

$y = x + Q(t)$

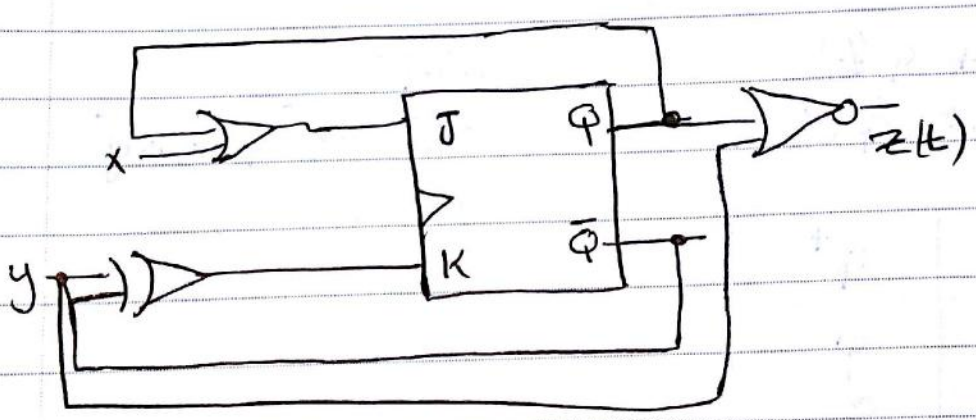
Mealy

smile for life

No. 3/8/2016



Example :- Determine the state table and diagram for the following circuit.



$N = 2 \Rightarrow x, y$

$M = 1 \Rightarrow Q$, 2 states

$K = 2$

$l = 1 \Rightarrow z(t)$

$2^3 * (2 + 1 + 1 + 1 + 2 * 1)$

$8 * 7$

No 3/8/2016

PS	IN			NS	OUT
φ	x	y	J K	φ^+	z
0	0	0	0 1	0	1
0	0	1	0 0	0	0
0	1	0	1 1	1	0
0	1	1	1 0	1	0
1	0	0	1 0	1	0
1	0	1	1 1	0	0
1	1	0	1 0	1	0
1	1	1	1 1	0	0

FF Input equation.

$J = x + \varphi(t)$

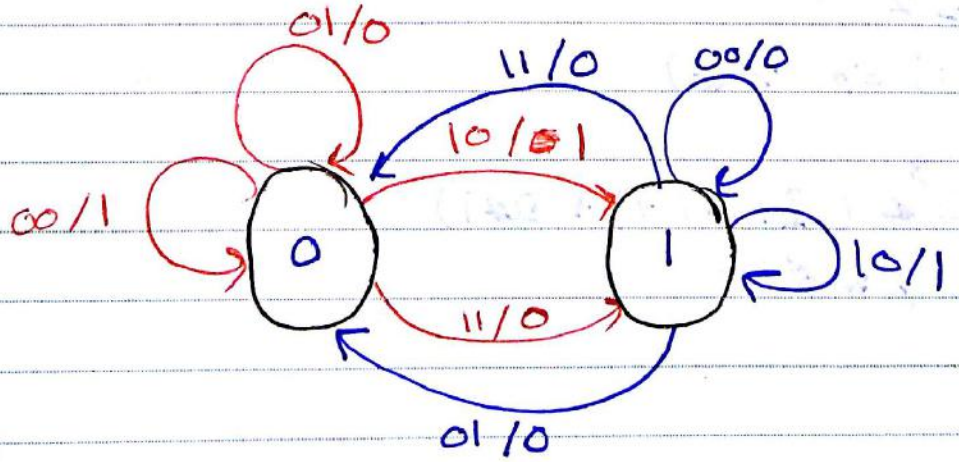
$K = y \oplus \overline{\varphi(t)}$

OUT

$z(t) = \overline{\varphi(t) + y}$

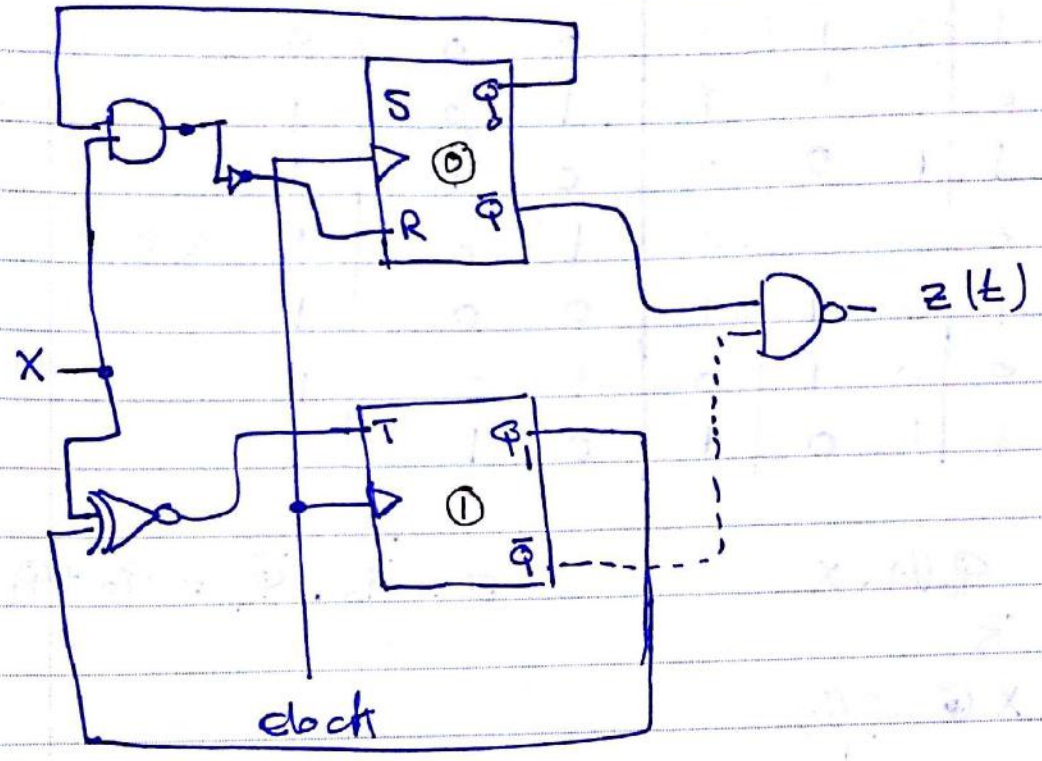
Mealy

J	K	φ^+
0	0	φ
0	1	0
1	0	1
1	1	$\bar{\varphi}$



No. 3/8/2016

Example:-



$N = 1 \Rightarrow X$

$M = 2 \Rightarrow Q_0, Q_1 \Rightarrow$ 4 states

$l = 1 \Rightarrow$

$K_0 = 2$

$K_1 = 1$

$2^3 * (1 + 2 + 1 + 2 + 2*1 + 1*1)$

$8 * 9$

No. 3/8/2016

PS		IN	FF0		FF1	NS		out
Q_1	Q_0	X	S	R	T	Q_1^+	Q_0^+	Z
0	0	0	0	1	1	1	0	0
0	0	1	0	1	0	0	0	0
0	1	0	0	1	1	1	0	1
0	1	1	1	0	0	0	1	1
1	0	0	0	1	0	1	0	1
1	0	1	0	1	1	0	0	1
1	1	0	0	1	0	1	0	1
1	1	1	1	0	1	0	1	1

$S = Q_0(t) \cdot X$

$R = \overline{S}$

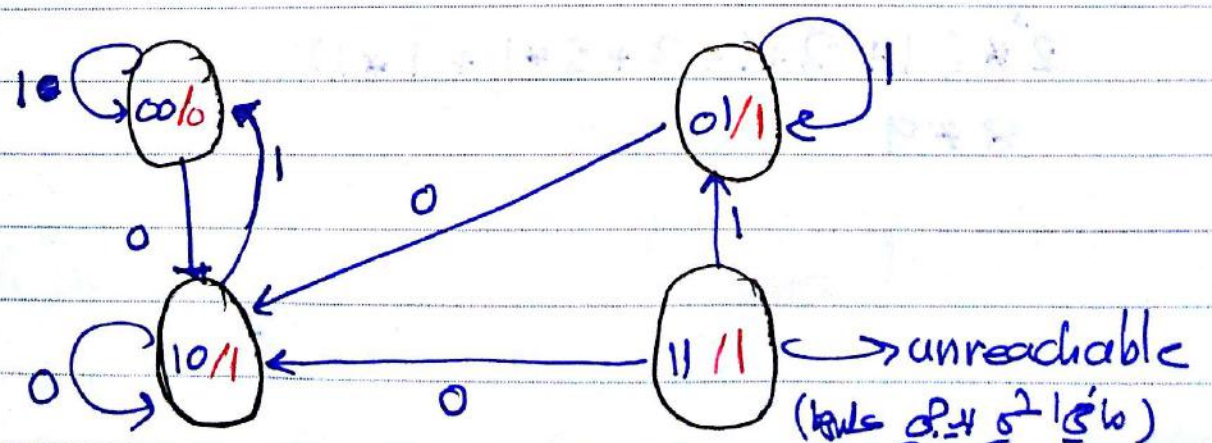
$T = X \oplus Q_1(t)$

$Z = \overline{Q_0} \cdot \overline{Q_1} = Q_0 + Q_1$

More

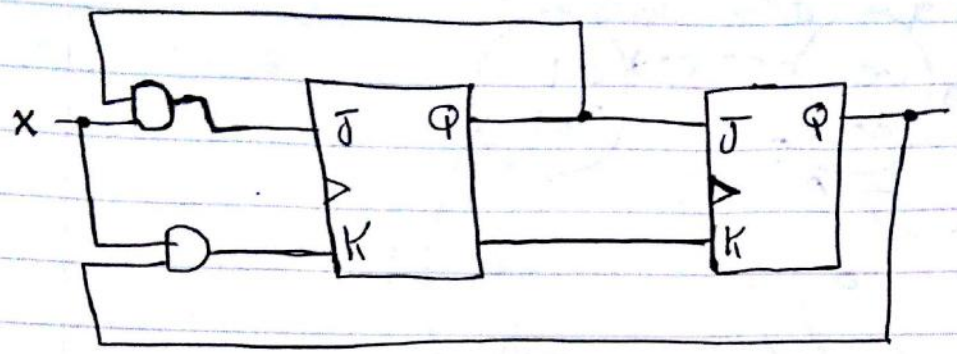
S	R	Q
0	0	Q
0	1	0
1	0	1
1	1	xx

T	Q+
0	Q
1	$\overline{Q}$



No. 3/8/2016

Example 2–



$N = 1 \Rightarrow x$

$M = 2$, $Q_0\ Q_1$

$FF = 2 \Rightarrow$ 4 states

	P.S		IN	FF$_1$	FF$_0$	N.S = output	
	Q_1	Q_0	x	J_1K_1	J_0K_0	Q_1^+	Q_0^+
①	0	0	0	0 1	0 0	0	0
	0	0	1	0 1	0 0	0	0
②	0	1	0	1 0	0 0	1	ϕ
	0	1	1	1 0	1 0	1	1
③	1	0	0	0 1	0 0	0	0
	1	0	1	0 1	0 1	0	0
④	1	1	0	1 0	0 0	1	0
	1	1	1	1 0	1 1	1	0

$J_0 = x \cdot Q_0$

$K_0 = x \cdot Q_1$

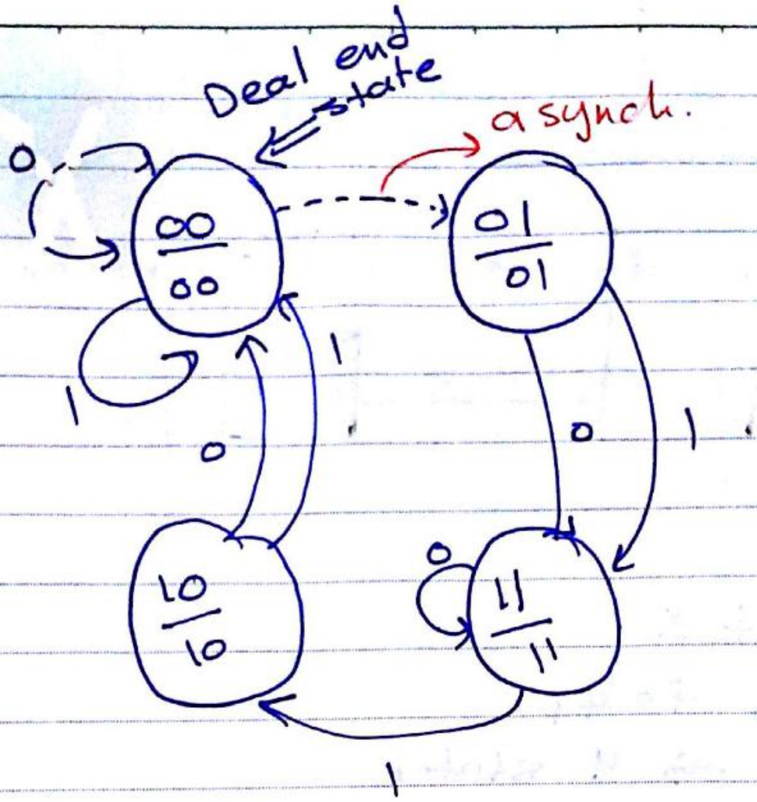
$J_1 = Q_0$

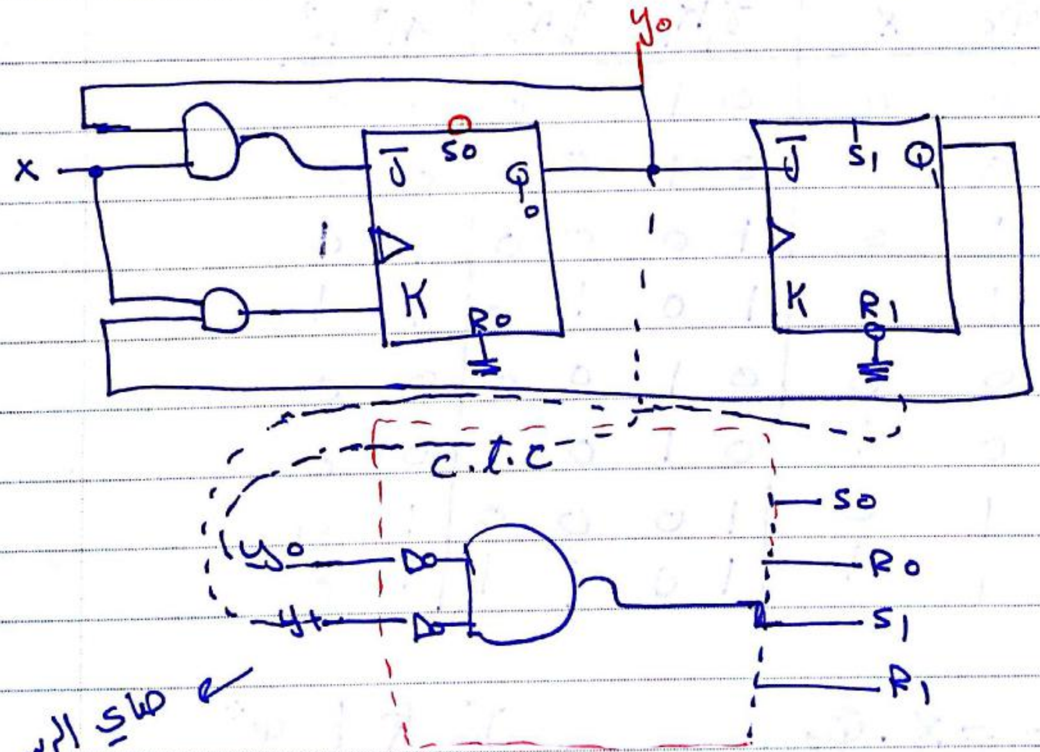
$K_1 = \bar{Q}_0$

J	K	Q^+
0	0	Q
0	1	0
1	0	1
1	1	$\bar{Q}$

smile for life

No 3/8/2016





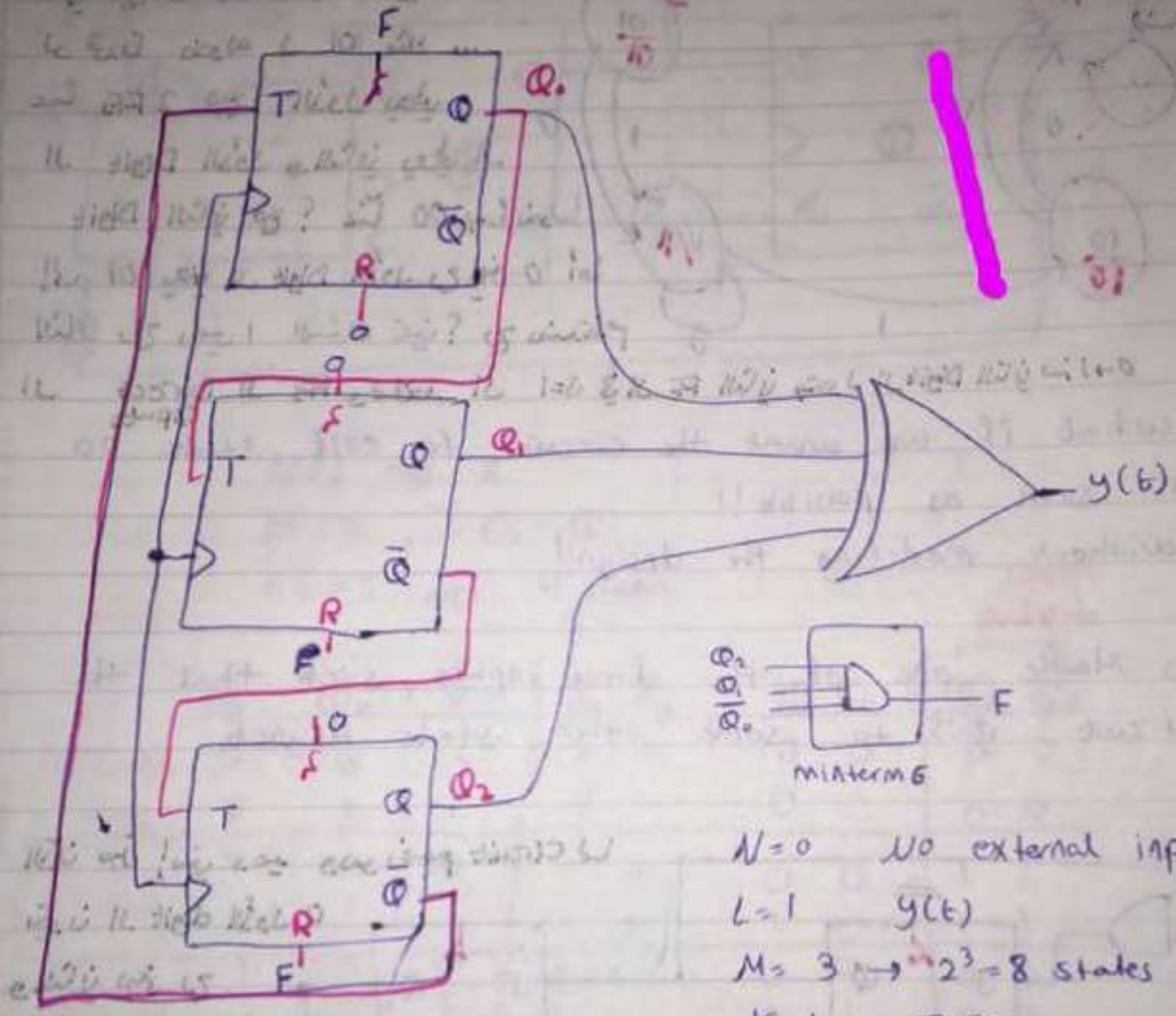
هاي الرسمة ←
شوفوا الصورة
أوضح
^^ :)

No. 3/8/2016

what if we want the circuit to exit state 00 as soon as possible !! without modifying the design!!

⇒ Make one of the direct inputs, such that the circuit goes to some other state Asynch.

Ex: S.C بدون external inputs



N = 0 No external inputs

L = 1 y(t)

M = 3 → $2^3 = 8$ states

K = 1 → T. FF.

F. Fs equations:

$T_2 = \bar{Q}_1$

$T_1 = Q_0$

$T_0 = \bar{Q}_2$

$y(t) = Q_2 \oplus Q_1 \oplus Q_0$ → Moore

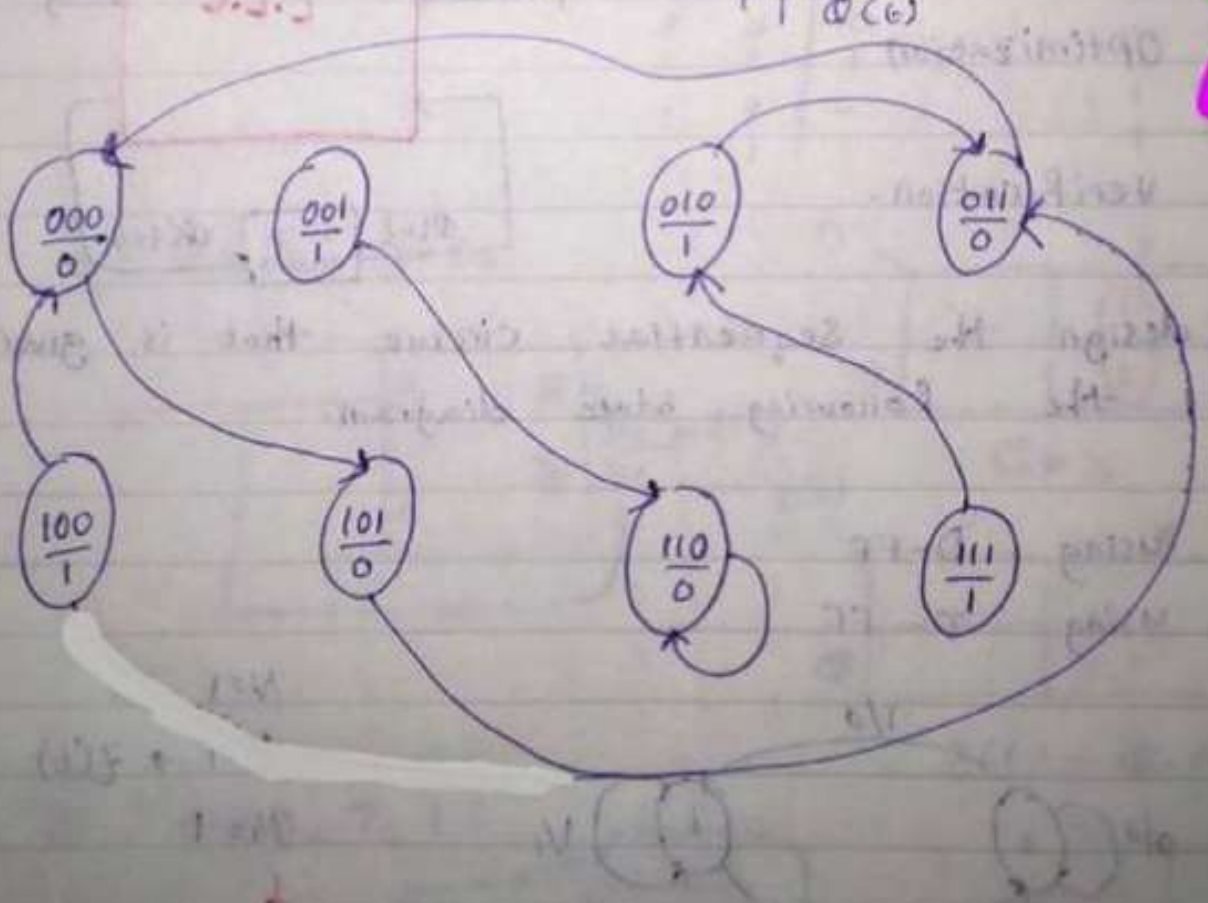
→ Unique want ↑

exit the state 110 to the state 001

⇒ State table

	present S				Next S			external output
	Q_2	Q_1	Q_0	T_2	T_1	T_0	Q_2^+	Q_1^+	Q_0^+	$y(t)$
000	0	0	0	1	0	1	1	0	1	0
001	0	0	1	1	1	1	1	1	0	1
010	0	1	0	0	0	1	0	1	1	1
011	0	1	1	0	1	1	0	0	0	0
100	1	0	0	1	0	0	0	0	0	1
101	1	0	1	1	1	0	0	1	1	0
110	1	1	0	0	0	0	1	1	0	0
111	1	1	1	0	1	0	1	0	1	1

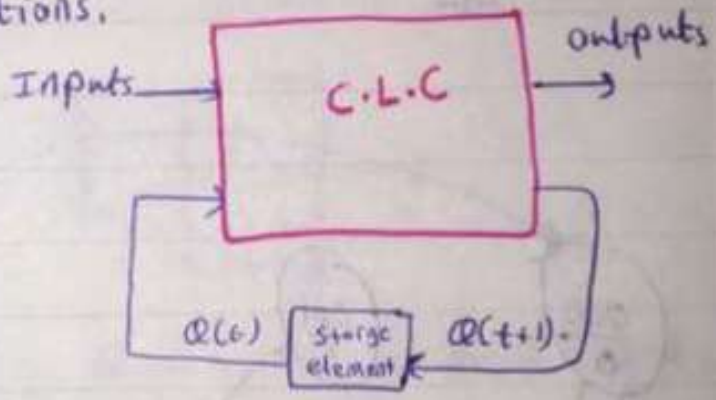
T	$Q_i^+(t)$
0	$Q(t)$
1	$\overline{Q}(t)$



5.6 Design of sequential circuits.

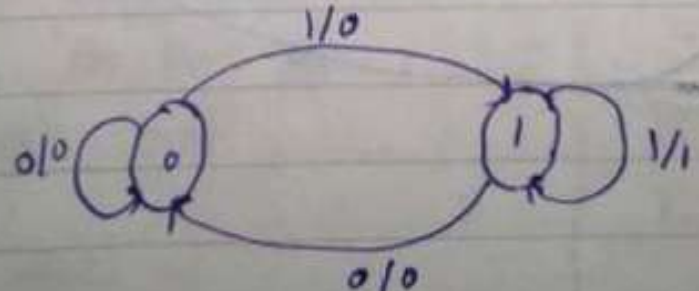
steps:-

① Sepecification / problem statement.
② Formulation → Derive state diagram or table
③ State assignment.
④ Determine FF input equations.
⑤ Determine output equations.
⑥ Optimization
⑦ Verification.



ex. design the sequential circuit that is given by the following state diagram.

① using D-FF
② using T- FF



N=1 → x
t=1 → z(t)
M=1
↓
M= Log 2 #states

Solution

⇒ State table

PS Q(t)	INPut X	N.S Q(t+1)	out z(t)	D D	T
0	0	0	0	0	0
0	1	1	0	1	1
1	0	0	0	0	1
1	1	1	1	1	0

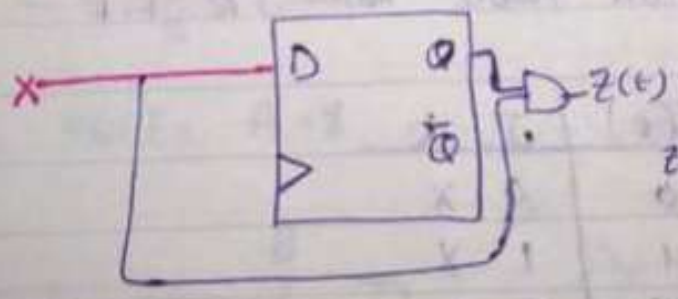
$D = Q^+$

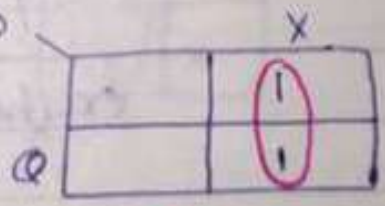
State diagram

Q	Q^+	D
0	0	0
0	1	1
1	0	0
1	1	1

Q	Q^+	T
0	0	0
0	1	1
1	0	1
1	1	0

① using D-FF

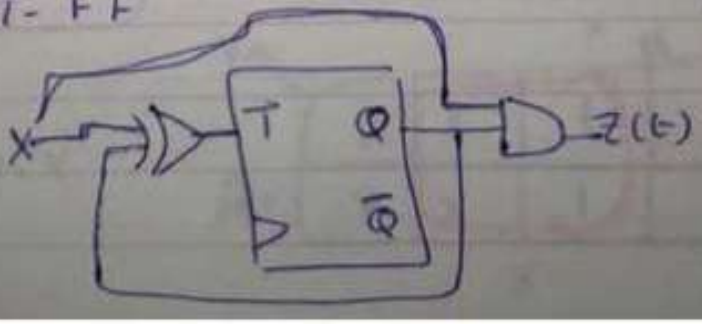


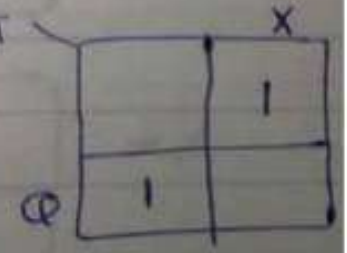


$D = X$

$z(t) = Q \cdot X$

② using T-FF





$T = Q \oplus X$

ex Design the following sequential circuit using JK F.F.

	P.S A	P.S B	IN X	N.S A^+	N.S B^+	OUT y(t)	J_A	K_A	J_B	K_B
00	0	0	0	0	0	0	0	X	0	X
	0	0	1	0	1	1	0	X	1	X
01	0	1	0	0	1	0	0	X	X	0
	0	1	1	1	0	1	1	X	X	1
10	1	0	0	1	0	1	X	0	0	X
	1	0	1	1	1	1	X	0	1	X
11	1	1	0	1	1	1	X	0	X	0
	1	1	1	0	0	1	X	1	X	1

excitation in table

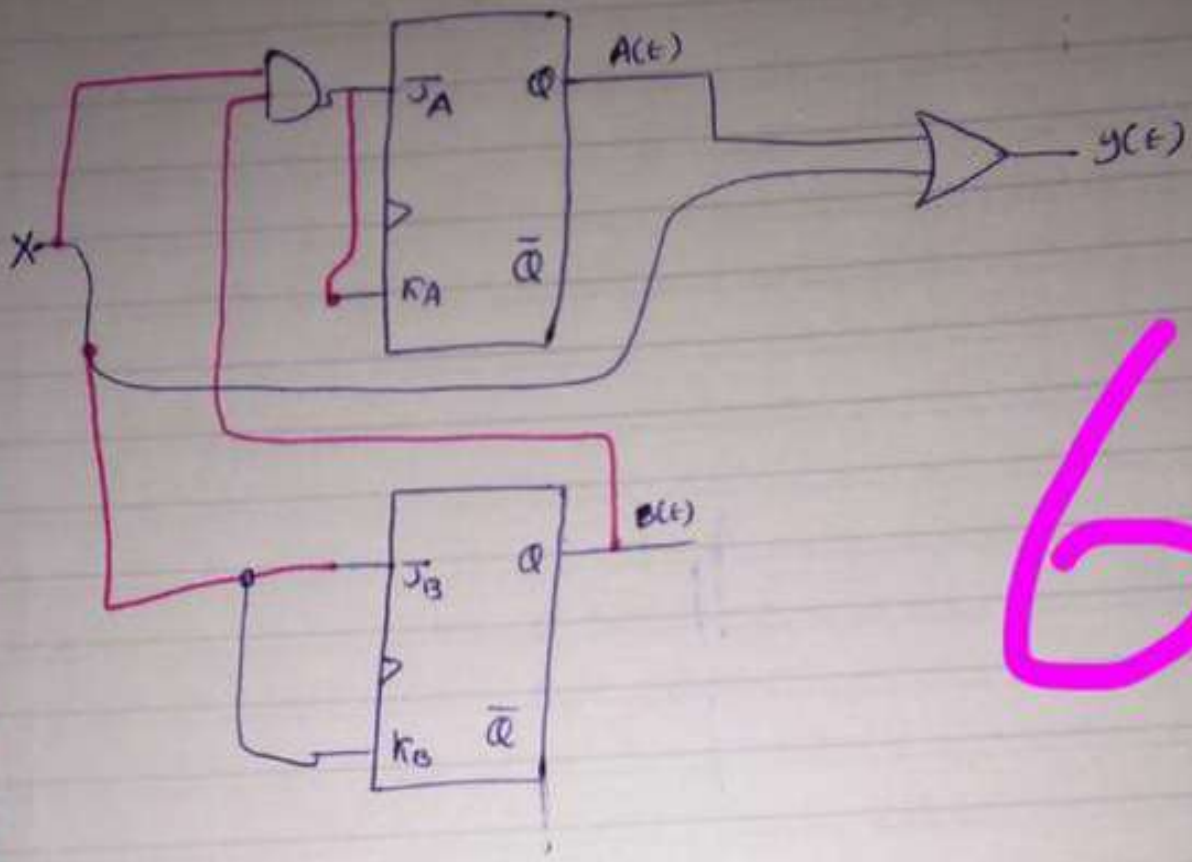
N = 1 (X)

L = 1 (y(t))

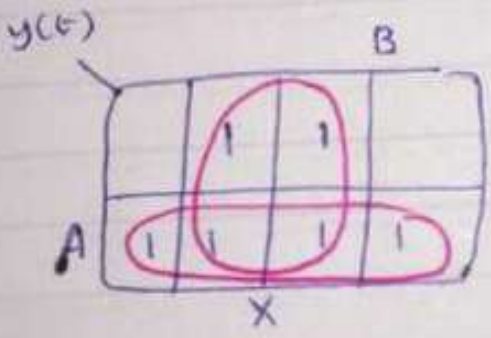
$M = 2 = \lceil \log_2 4 \rceil = 2$ FF → A, B

excitation table for JK F.F

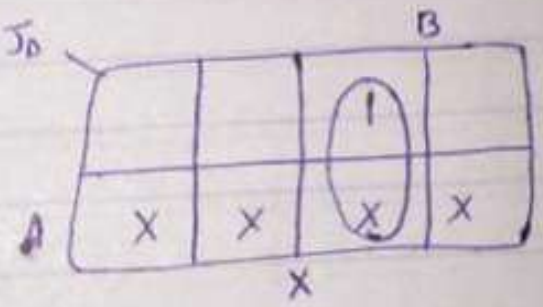
Q(t)	$Q^+(t)$	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0



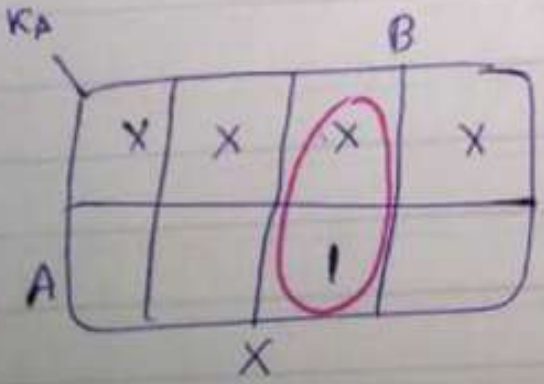
6



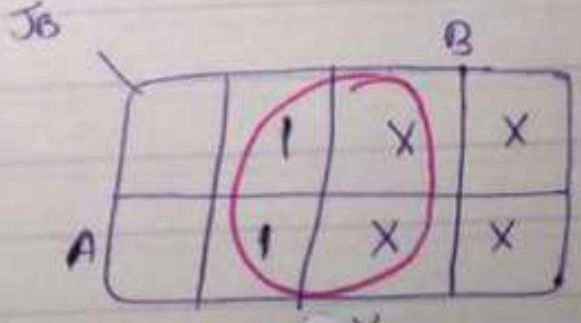
$y(t) = A + X$



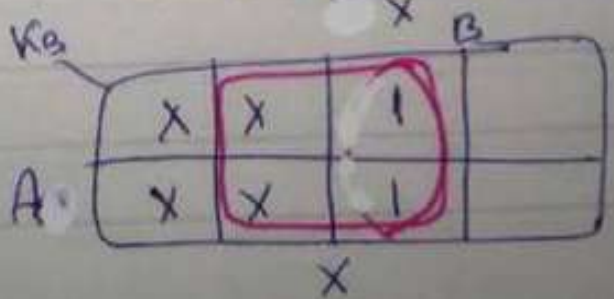
$J_A = B(t)\ X$



$K_A = B(t) \cdot X$



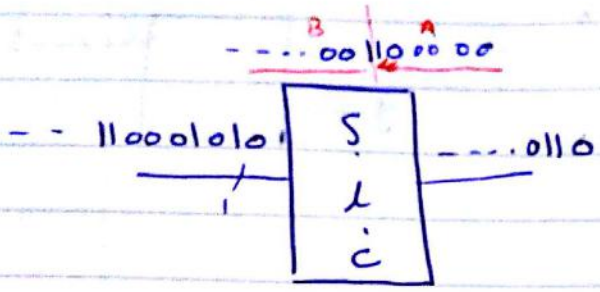
$J_B = X$



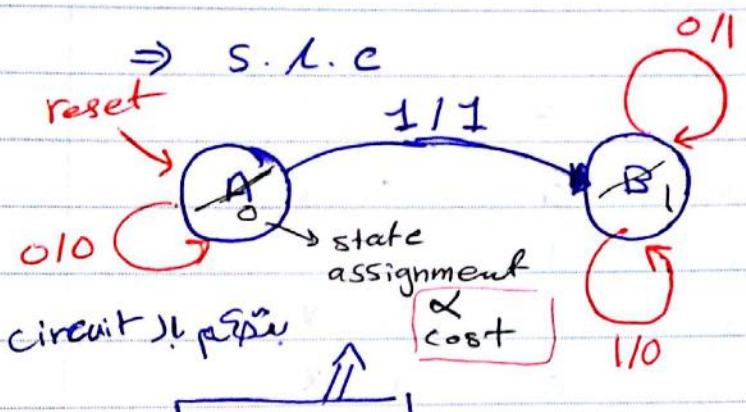
$= X$

No. 8/8/2016

* Example:- Design a serial / sequential 2's complementor

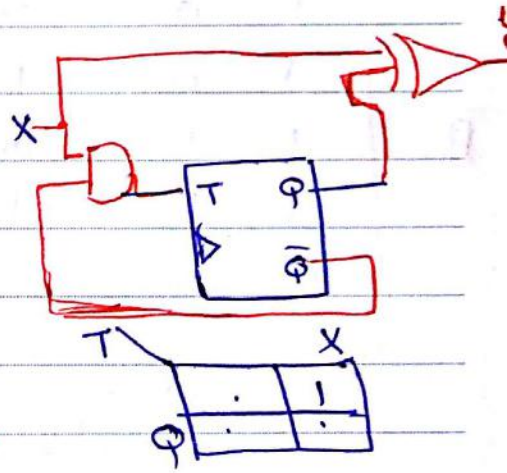


we need to remember wether the 1st one has arrived !!

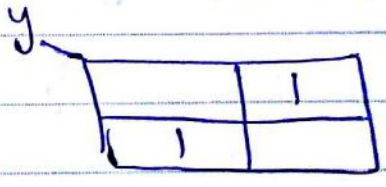


two states ⇒ 1 FF ⇒

Q	X	Q+	y	T
A	0	A	0	0
A	1	B	1	1
B	0	B	1	0
B	1	B	0	0



$T = \bar{Q} \cdot X$

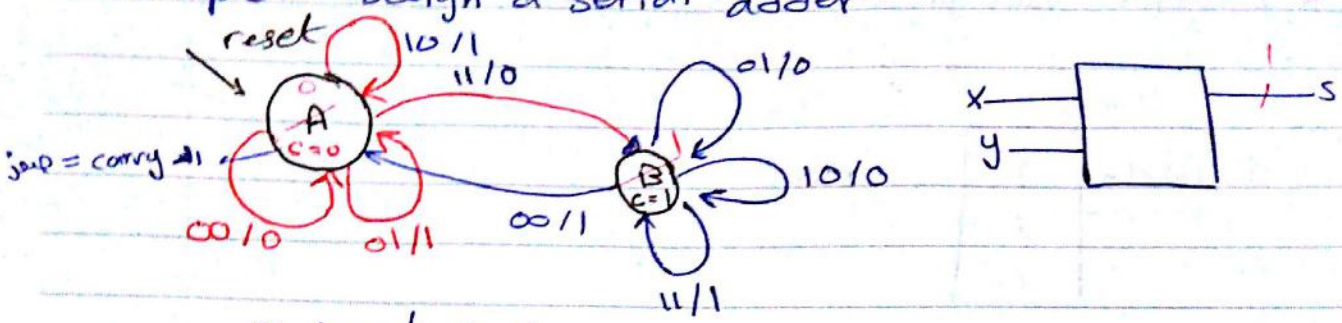


$y = Q \oplus X$

smile...

No 3/8/2016

Example 1- Design a serial adder



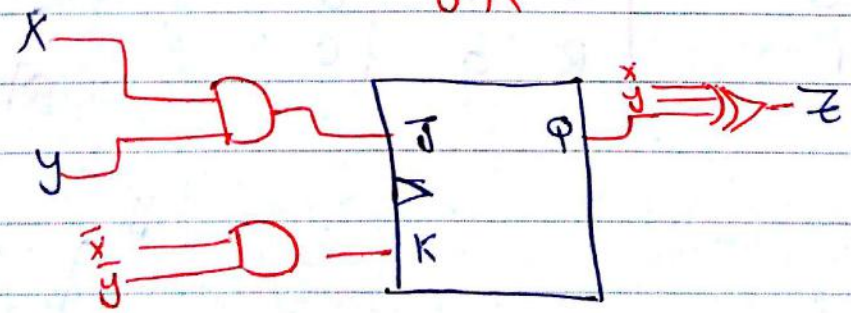
2 input → x, y

2 states ⇒ 1 FF

1 output ⇒ Z

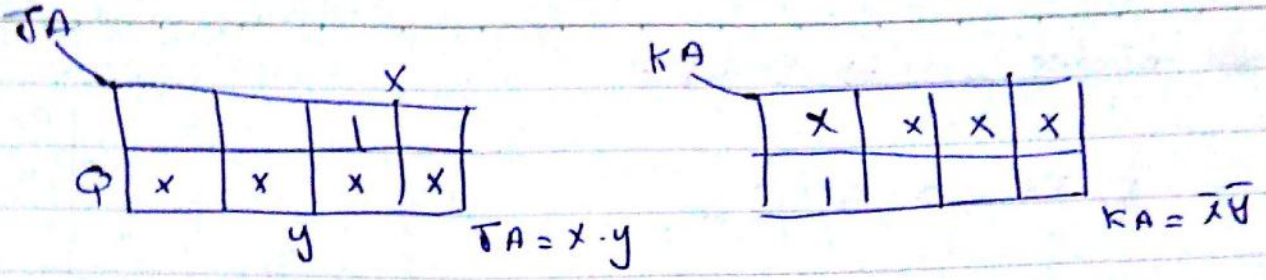
	Q	x	y	Q^+	Z	J	K
A	0	0	0	0	0	0	X
	0	0	1	0	1	0	X
	0	1	0	0	1	0	X
	0	1	1	1	0	1	X
B	1	0	0	0	1	X	1
	1	0	1	1	0	X	0
	1	1	0	1	0	X	0
	1	1	1	1	1	X	0

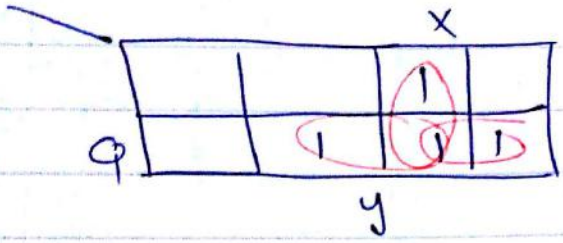
JK

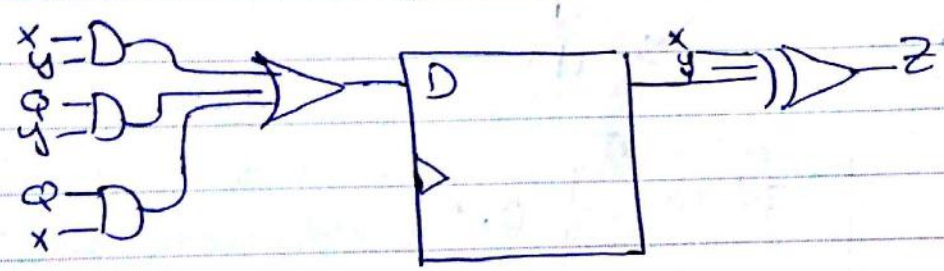


smile

No. 3/8/2016







5.7 Counters :-

It's a sequential (synch. / Asynch.) circuit that goes ~~wintks~~ into acertain counting order.

EX:- Desing a counter that goes through the following counting order.

1, 2, 0, 3, 1, 2, 0, 3, 1, 2, 0, 3, ---- ----

smile

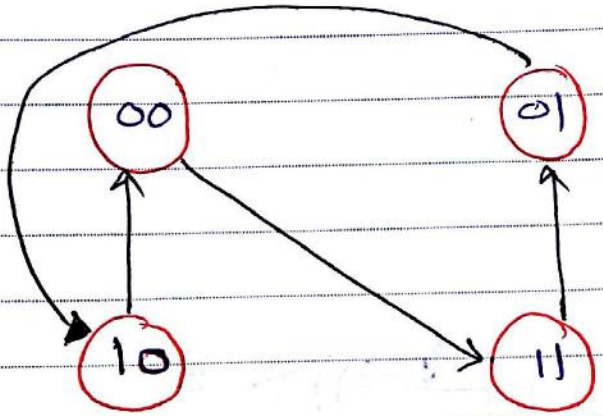
No. 8/8/2016

states. = 4 ⇒ 2 FFs

D FF ⟹ Q_1 Q_0

Inputs ⟹ No Input

outputs ⟹ Q_1 Q_0

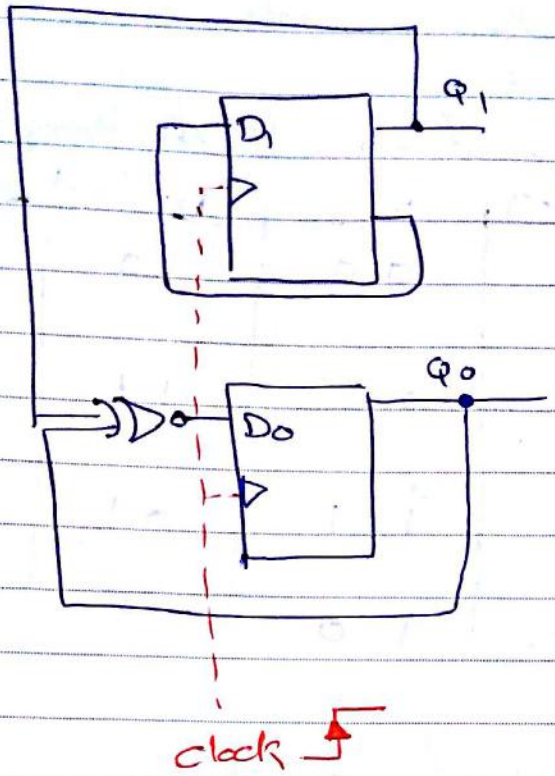


PS		NS = OUT		
Q_0	Q_1	Q_1^+	Q_0	D_1 D_0
0	0	1	1	
0	1	1	0	
1	0	0	0	
1	1	0	1	

$D_1 = \overline{Q_1}$

$D_0 = Q_1 \odot Q_0$

No. 8/8/2016



Ex :- Design a two-bit counter that has one input x and operates as follows:-

① x = 0 → hold the count

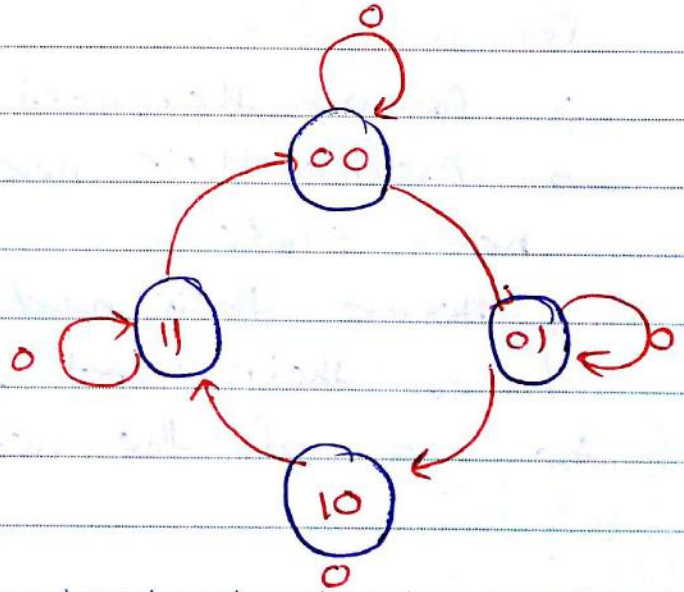
② x = 1 → count normally

0, 1, 2, 3, 0, 1, 2, 3, - - -

4 states → 2 FF

1 input → x

outputs → Q_1, Q_0



smile

No 8/8/2016

5.8 Design issues.

#FF = # states

(A) state assignment ⇒ affects complexity

	counting	gray	Random	one-hot-key
A	0 0	0 0	1 0	0 0 0 1
B	0 1	0 1	1 1	0 0 1 0
C	1 0	1 1	0 1	0 1 0 0
D	1 1	1 0	0 0	1 0 0 0

(B) Design with unused states.

For some circuits, the number of required states, might be less than the available states!!

* How to deal with unused state ?!

For unused states :-

→ Assume their next state as don't care.

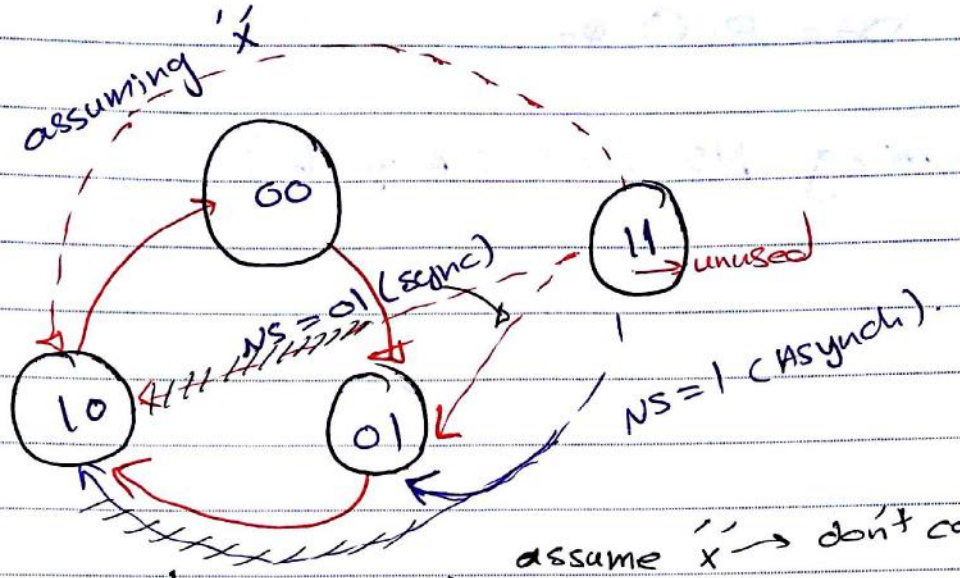
⇒ Assume their next state to be one of next state

→ Assume their next state to be don't and force their next state to change asynch. to one of the used states.

No. 8/8/2016

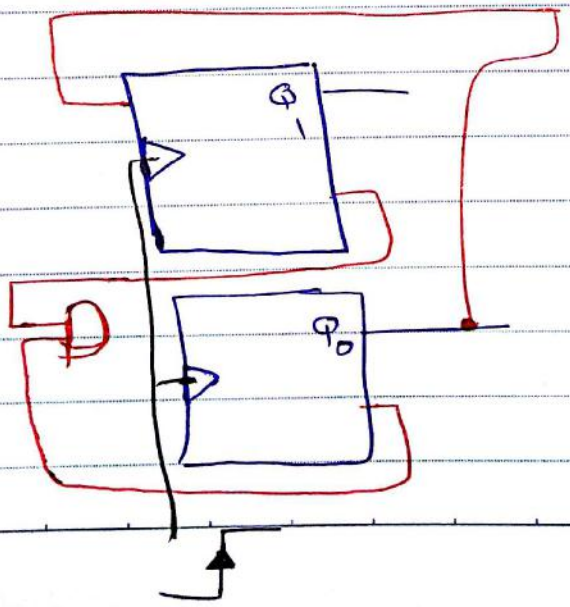
Example:- Counter 0, 1, 2, 0, 1, 2, 0, 1, 2, ——

3 state → 2 FFs → Q_1 Q_0



assume 'x' → don't care

Q_1 Q_0	Q_1^+ Q_0^+	D_1 D_0	
0 0	0 1	0 1	$D_0 = \overline{Q_0}.\overline{Q_1}$
0 1	1 0	1 0	
1 0	0 0	0 0	~~$D_1 = \overline{Q_1}.\overline{Q_0}$~~
1 1	x x	x x	$D_1 = Q_0$



smile

No. 9/8/2016

~ بسم الله ~

State Equivelence ...

⇒ # states ∝ # FFs

FFs = $\lceil \log_2 \#\text{states} \rceil$

⇒ The cost of the sequential circuit depends on # FFs

⇒ can we reduce # FFs ?!

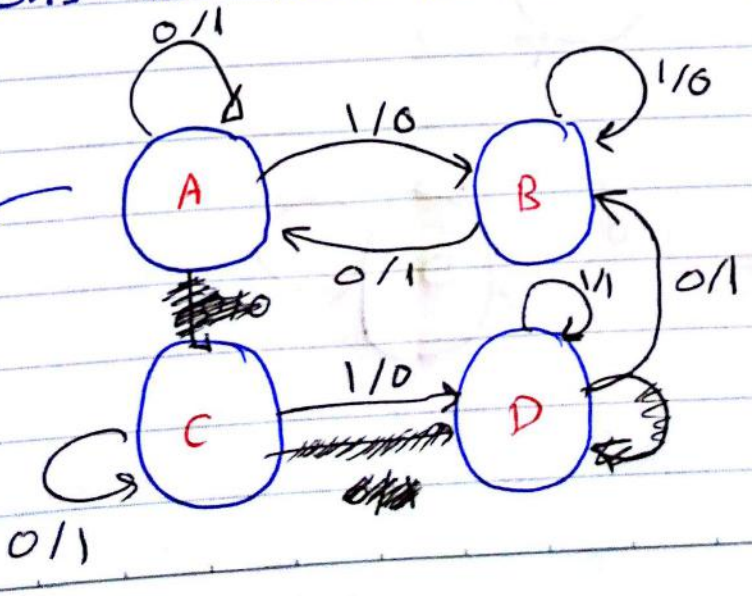
Yes if #states is reduced !!

⇒ In some designs, we may have equivelent stats that can be combined ...

⇒ Equivelent states

Two states are equivelent if for all Input Combinations their next state and output are the same !!

Example ←



smile...

No. 9/8/2016

A / B

A { 0 A 1
  { 1 B 0 $A \equiv B$

B { 0 A 1
  { 1 B 0

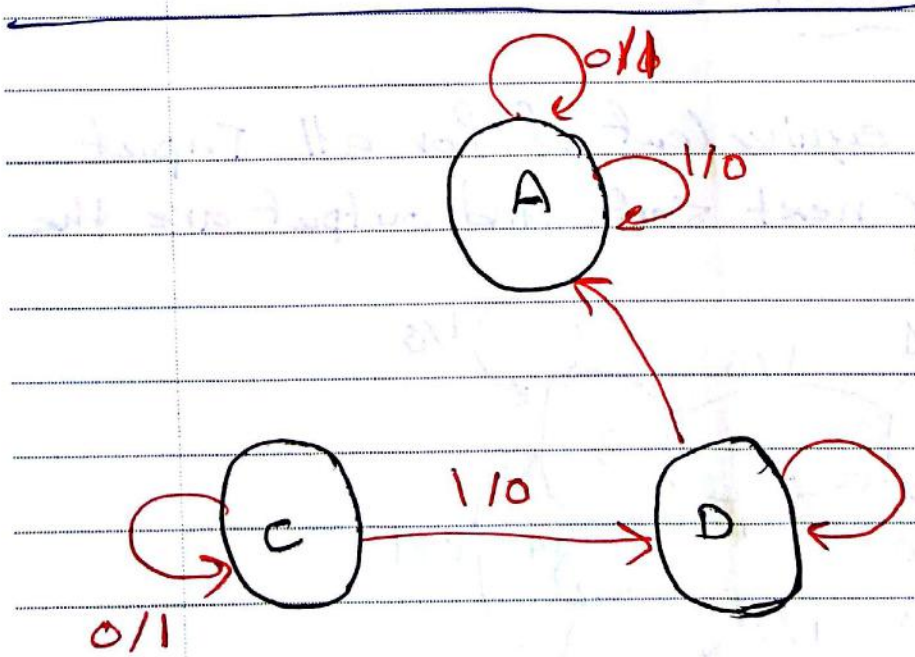
A / C

{ 0 A 1
{ 1 B 0 $A \not\equiv C$

{ 0 C 1
{ 1 D 0

{ 0 A 1
{ 1 B (0)

{ 0 B/A 1 $A \not\equiv D$
{ 1 D (1)



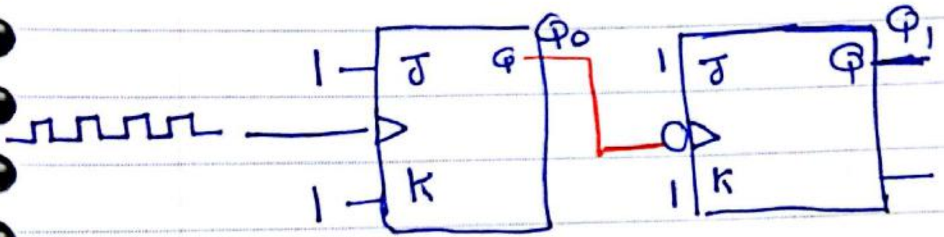
No. 9/8/2016

CH. 7 Registers and counters...

7.1 counters

→ synch. ⇒ All FFs are connected to the same clock.

→ A synch.

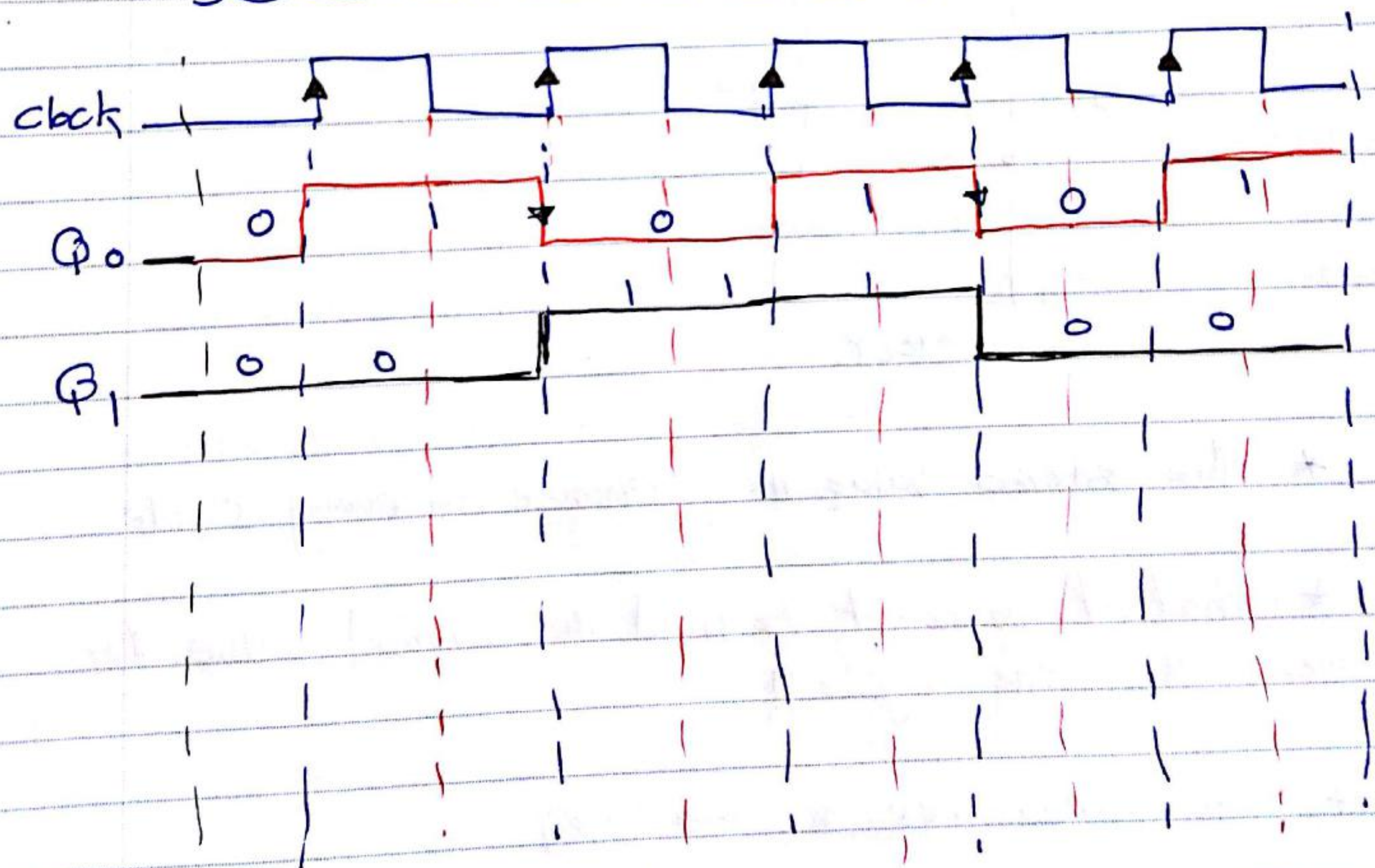


⇒ Asynch

لل clock لل FF

لأنها مش نفسها

لل clock لل FF يالي



0 → 1 → 2 → 3 → 0 → 1

2-bit counter asynch.

modulo - 3 counter

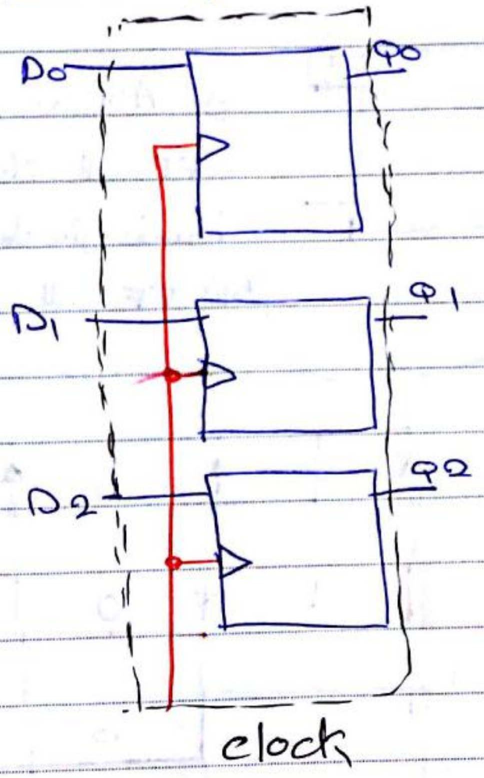
No. 9/8/2016

7.2 Registers ...

* is a group of storage elements that holds stored bit

* n-bit register → n storage elements

* 3 bit Register



* The stored value may change on every cycle.

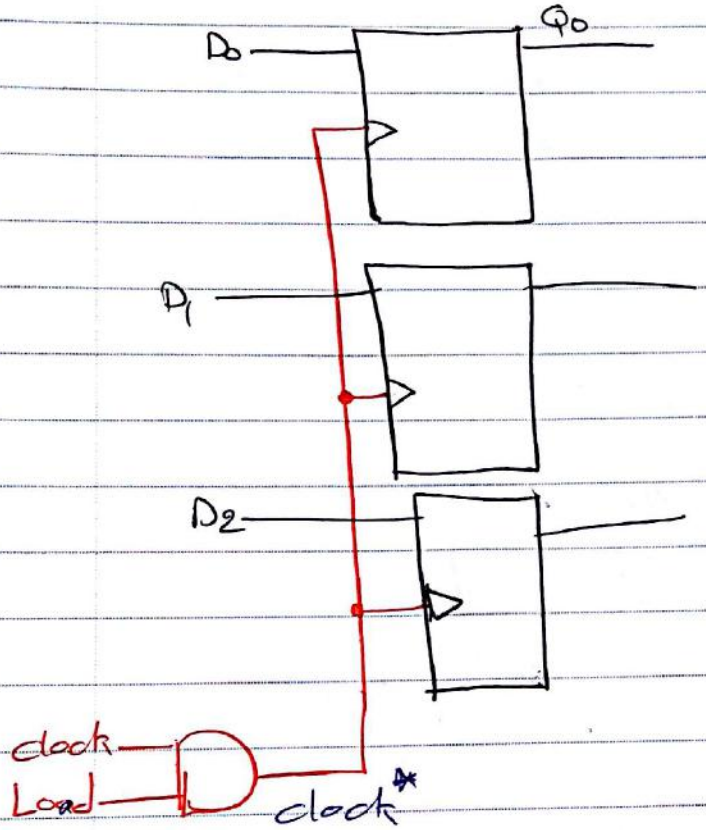
* What if we want to hold the stored value for more than one cycle!!

* Two approaches : →

Scanned by CamScanner

No. 9/8/2016

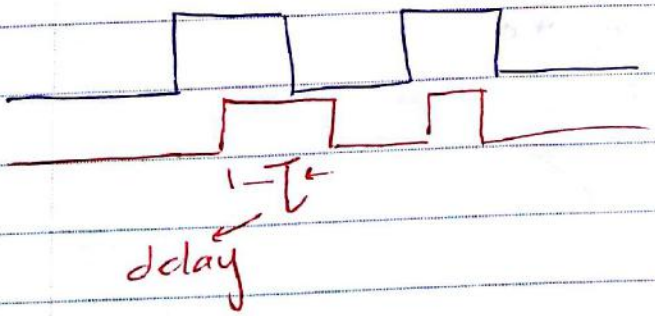
① gated clocks



Load = 0 ⇒ clock* = 0 → no edges → no change

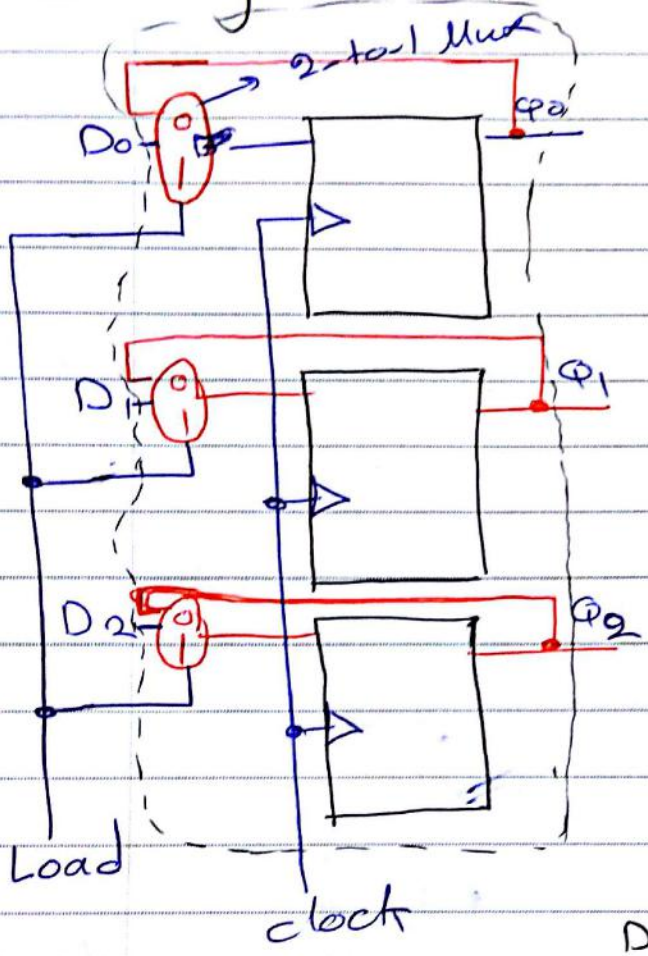
Load = 1 ⇒ clock* = clock → change is allowed

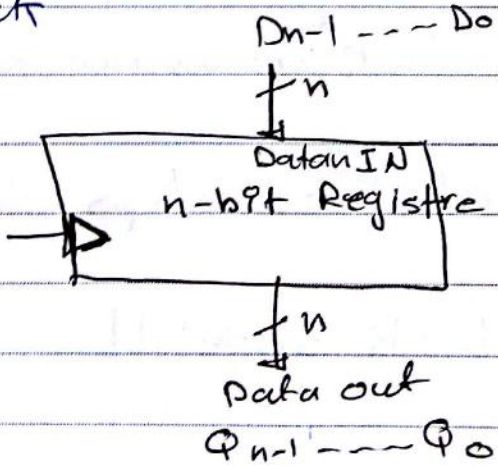
* problem → clock skew !!



No. 9/8/2016

② using Muxes





⇒ n-bit Register

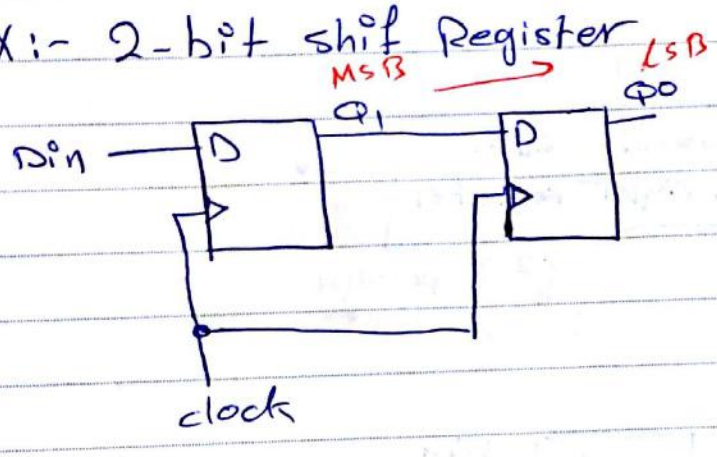
No. 9/8/2016

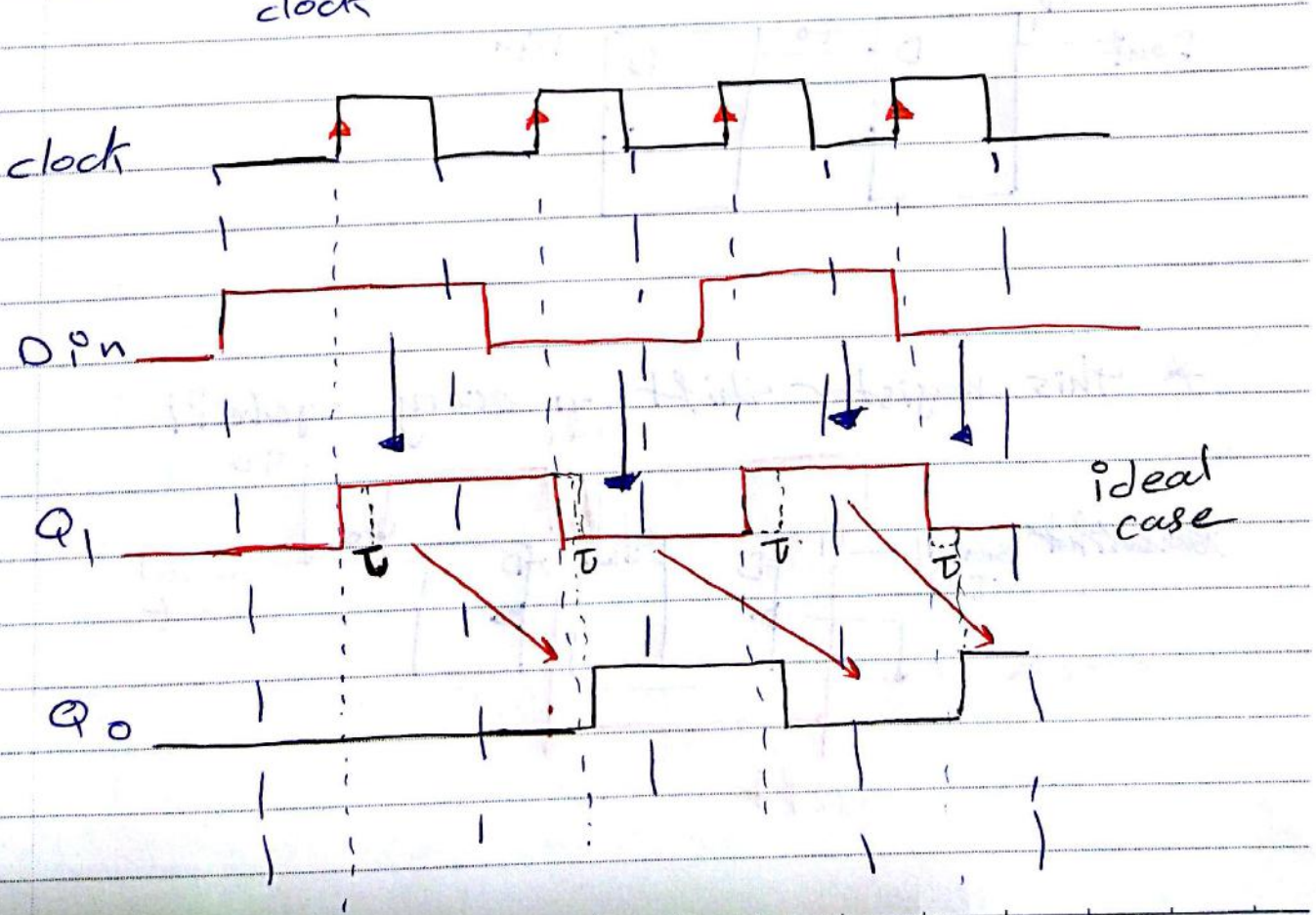
7.3 Shift Register

It's a register with it's FFs connected in cascade in away that allows shifting/moving the stored value toward the MSB or LSB position.

shift left ↙ ↘ shift right

EX:- 2-bit shift Register

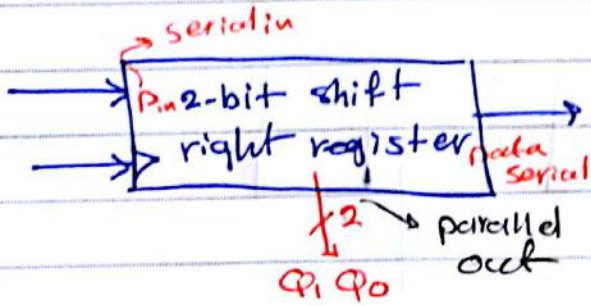


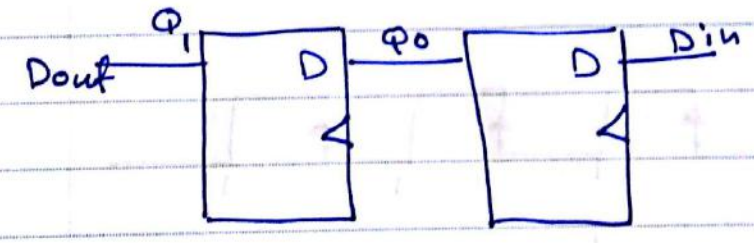


smile

No 9/8/2016

↓	Q_1	Q_0
1	0	0
2	1	0
3	0	1
4	1	0





* this register shift on every cycle?!